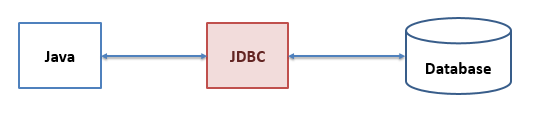
**JDBC**

* Java Data Base Connectivity is a Java API that helps us to achieve the connectivity between Java & Database
* If we have a Web Application & if it has a DB, then we have to interact with DB to read / modify the data. JDBC helps us to achieve this & in the world of Java, **JDBC is the One & Only API that helps us to interact with DB**.

****

**Advantages of JDBC**

1. Since JDBC API is built upon Java it eventually inherit the advantages of Java
2. JDBC is DB Independent
3. Using JDBC we can interact with Multiple DB simultaneously
4. You can achieve High Performance by using JDBC with the help of PreparedStatements & CallableStatements
5. JDBC supports Stored Procedures

**Interview Question:** What are the advantages we get with Java / Why Java?

**JDBC Prerequisites:**

1. Install the DB Server (MySQL 5.5)
2. Create a DB (by name BATCH-CODE\_DB)
3. Create a table in the above DB (by name STUDENTS\_INFO)
4. Insert some records into the above table

**DB Queries:**

**CREATE** DATABASE batch-code\_db;

**CREATE** **TABLE** students\_info

( regno **INT**(50) **NOT** **NULL**,

firstname **VARCHAR**(50),

middlename **VARCHAR**(50) **DEFAULT** ‘Not Avilable’,

lastname **VARCHAR**(50),

**PRIMARY** **KEY** (regno)

);

**INSERT** **INTO** students\_info **VALUES** (1, ‘XYZ’, ‘NA’, ‘ABC’);

**INSERT** **INTO** students\_info (regno, firstname, lastname) **VALUES** (2, ‘Praveen’, ‘D’);

**Assignment 1:**

* Create table by name “guardian\_info” which will have the following structure

guardian\_info

* regno int(10)(PK)
* gfirstname varchar(50)
* gmiddlename varchar(50)
* glastname varchar(50)
* Create a one more table by name “students\_otherinfo” which will have the following structure

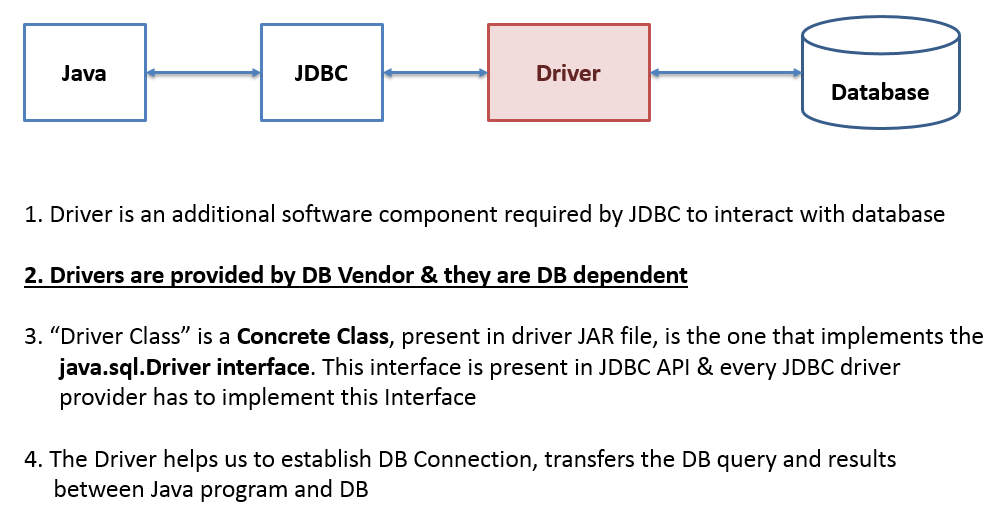
students\_otherinfo

* regno(10)(PK)
* isadmin(1)
* password(10)
* Insert regno 1 to 5 data into above tables. While inserting the data into “students\_otherinfo” make regno 1 isadmin flag as “Y” and rest as “N”

**Necessary Steps to work with JDBC**

1. Load the JDBC **Driver**
2. Get the **DB Connection** via **Driver**
3. Issue **SQL Queries** to via **DB Connection**
4. Process the **Results** returned by **SQL Statements**
5. Close all the **JDBC Objects**

**Drivers**

****

**Steps to Configure the DB Driver**

**Step 1 – Configure Project Build Path:**

* Download the MySql Driver Zip file & extract it to some location
* Go to Eclipse, right Click on the Project, select Build Path ---> Configure Build Path
* Go to “libraries” tab, Click on the “Add External JARs” button
* select the Driver JAR file, click on “Open” button
* Click on OK button

**Note:** JAR (Java Archive) is a Collection of source files (Java Classes) & other necessary resource files (XML, Properties, Text Files, etc.). Like Zip file, JAR file is used to transfer the set of programs from one computer to an another

**Step 2 – Load the Driver Class into Program:**

There are two ways to load the Driver Class,

1. Driver driver = **new** com.mysql.jdbc.Driver();

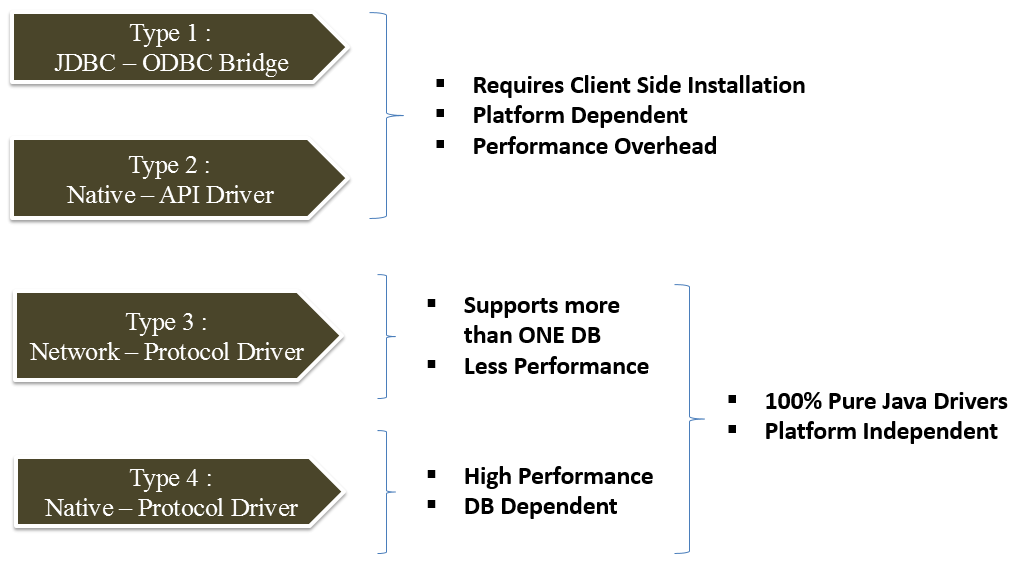
DriverManager.*registerDriver*(driver);

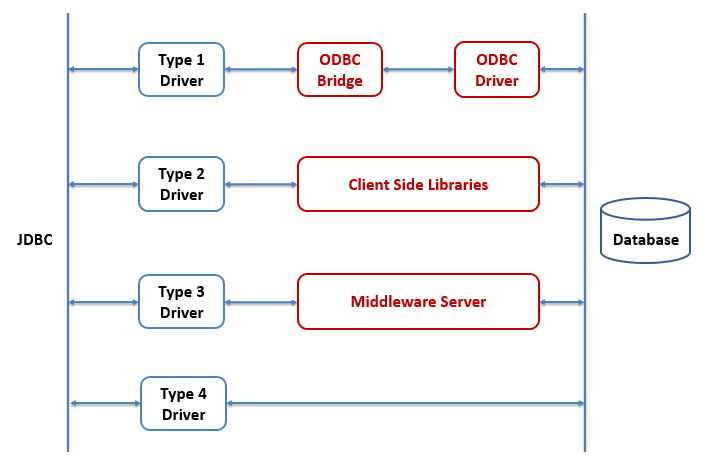
1. Class.forName("**com.mysql.jdbc.Driver**");

* The most common approach to register a driver is to use Java's Class.forName() method which dynamically loads the driver's class file into memory & automatically registers it.
* The main advantage with Class.forName is, it can accept the class name as a String argument so that we can pass the driver class name dynamically. But if we create an instance of a driver class using new operator, then driver class name can't be changed dynamically.

**Drivers Types**

There are 4 different types of driver





**Questions:-**

* Let's say your application design is a two-tier architecture where the database client operates as a standard Java application. What driver type are best suited to this type of application design?
* Which driver type is best suited to a single-database environment?

**Explain 1st Advantage of JDBC – DB independent**

**DB URL (Uniform Resource Locators)**

* DB URL, uniquely identifies the database in a network (internet / intranet)
* The structure of DB URL is

**<Protocol> : <Subprotocol> : <Subname>**

**1. Protocol: -** In case of JDBC, it is always “**jdbc”**

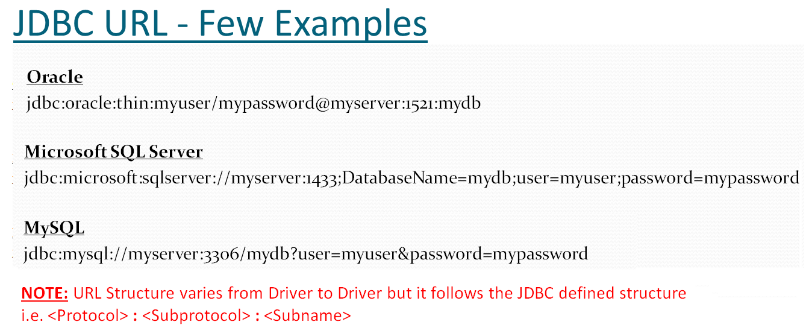
**2. Subprotocol: -** It is the name of the database connectivity mechanism. This information is provided in the driver manual

**3. Subname: -** It consist of,

1. Host Name (computer name or IP Address in which DB Server Application is installed)
2. Port on which DB Server application is installed
3. DB name
4. User ID & Password to connect to DB

**Port:**

It uniquely identifies an application in an operating system. The default port number for MySQL DB Server is 3306

****

**java.sql.DriverManager**

* This class is part of JDBC API & as the name implies, it manages the drivers. It helps us to establish the connection to DB & it requires following two critical information to establish the connection to DB

1. DB URL (to which DB it has to establish the connection)
2. Driver Class (using which Driver)

* DriverManager is a concrete class present in JDBC API & consist of only one constructor which is private default in nature and hence it cannot be inherited or instantiated directly. It exposes lot of methods to outside world & they are public static in nature.
* DriverManager’s getConnection() method helps us to establish a connection to the DB. The return type of getConnection() method is object of type java.sql.Connection interface.
* Connection Object is an Object representation of the **physical database connection** that can be used by a Java program to communicate with database.
* DriverManager has 3 overloaded getConnection() methods

1. **public** **static** Connection getConnection(String url)

**throws** SQLException

String dbUrl = "jdbc:mysql://localhost:3308/college?user=j2ee&password=j2ee";

Connection con = DriverManager.*getConnection*(dbUrl);

1. **public** **static** Connection getConnection(String url, String user, String password) **throws** SQLException

String dbUrl = "jdbc:mysql://localhost:3308/college";

String userId = "j2ee";

String password = "j2ee";

Connection con = DriverManager.*getConnection*(dbUrl, userId, password);

1. **public** **static** Connection getConnection(String url, java.util.Properties prop) **throws** SQLException

String dbUrl = "jdbc:mysql://localhost:3308/college";

String fileAndPathNm = "D:\\j2ee\\db.properties";

FileInputStream stream = **new** FileInputStream(fileAndPathNm);

Properties prop = **new** Properties();

prop.load(stream);

Connection con = DriverManager.*getConnection*(dbUrl, prop);

**Contents of db.properties file:**

**JDBC Statements**

* JDBC Statements send SQL queries to DB and retrieve the data from database.
* There are 3 different types of JDBC Statements

1. java.sql.Statement
2. java.sql.PreparedStatement
3. java.sql.CallableStatement

* Once you've created a JDBC Statement object (any of the above type), then we must invoke any one of the below method to issue SQL queries to DB

1. **int executeUpdate() :**
   * This method is used to execute any SQL Query “**other than Select**”
   * This method returns the numbers of rows affected count in the form of integer
2. **ResultSet executeQuery() :**
   * This method is used to execute **“ONLY Select SQL Query”**
   * This method returns a DB results in the form of ResultSet object
3. **boolean execute() :**

* This method is used to execute **“ANY SQL queries including Select”**
* This method returns true if the result is of type DB Results & returns false if it is a integer count
* If we use this method then we must use the methods getResultSet() or getUpdateCount() to retrieve the result.

**java.sql.Statement**

* Statement object is used to execute **Static SQL Queries**
* SQL Queries without conditions or SQL Queries with hard coded condition values are known as static SQL queries. For example,
* select \* from ABC;
* create database DB\_NAME;
* select \* from ABC where X = 1;
* insert into ABC values (1, 'Praveen');

Where ABC = Table Name

**java. sql.PreparedStatement**

* PreparedStatement object is used to execute Dynamic SQL queries
* SQL Queries which must have conditions & condition values get decided at runtime are known as Dynamic SQL Queries

For Example,

1. select \* from ABC where X = ? and Y = ? ;
2. select \* from ABC where X = 1 and Y = ? ;
3. insert into ABC values (?, 'Praveen');

Where ABC = Table Name

NOTE : Dynamic SQL Query MUST Contain One / More Question Marks

* PreparedStatements **MUST** be used with query parameters (?) & these query parameters needs to be set using proper setXXX() method before executing the dynamic SQL query
* They are also used in a situation where same SQL queries is executed many times
* PreparedStatements are also known as “Precompiled Statements” & PreparedStatements helps us to achieve high performance

**java.sql.CallableStatements:**

* CallableStatement object is used execute **Stored Procedures**
* A stored procedure is a group of SQL queries that perform a particular task. As its name implies, they are stored at DB side and they help to achieve reusability

**Stored Procedure 1:-**

**DELIMITER** $$

**CREATE** **PROCEDURE** GetAllStudents()

**BEGIN**

**SELECT** \* **FROM** students;

**END** $$

**DELIMITER** ;

**call** GetAllStudents();

**Stored Procedure 2:-**

**DELIMITER** $$

**CREATE** **PROCEDURE**

GetStudentInfo(**IN** inputRegNo **INT**)

**BEGIN**

**SELECT** \* **FROM** students

**WHERE** regno = inputRegNo;

**END** $$

**DELIMITER** ;

**call** GetStudentInfo(1);

**Stored Procedure 3:-**

**DELIMITER** $$

**CREATE** **PROCEDURE** StudentsUpsert

( **in** in\_regno **int**,

**in** in\_fname **varchar**(50),

**in** in\_mname **varchar**(50),

**in** in\_lname **varchar**(50)

)

**BEGIN**

**DECLARE** regnoCount **int**;

**SELECT** **COUNT**(regno) **INTO** regnoCount

**FROM** students

**WHERE** regno = in\_regno;

**IF** regnoCount > 0 **THEN**

**UPDATE** students

**SET** firstname=in\_fname, middlename=in\_mname, lastname=in\_lname

**WHERE** regno = in\_regno;

**ELSE**

**INSERT** **INTO** students

**VALUES** (in\_regno, in\_fname, in\_mname, in\_lname);

**END** **IF**;

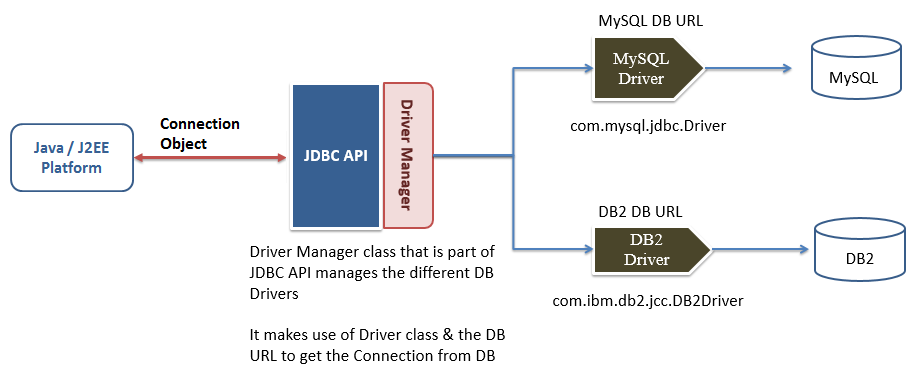
**END**$$

**DELIMITER** ;

**Assignment 2:**

* Create a stored procedure by name “StudentUpsert2” in such a way that it should take regno, student first name, student middle name, student last name, guardian first name, guardian middle name, guardian last name, isadmin and password information as input arguments.
* Check if input register number is present in “students\_info” table
* If present then update the corresponding record in “students\_info”, “gurardian\_info” & “students\_otherinfo” tables based on the incoming data
* If not present then insert a new record into above tables

**Explain this image but don’t give i.e. 3rd Advantage of JDBC – helps us to interact with Multiple DB’s. Also explain the 4th & 5th Advantage as well**



**Processing the Result**

* Whenever we issue SQL Queries to DB there are two kinds of results expected out of DB server

1. No. of Rows affected count

2. DB Results whenever we issue Select SQL queries

* In JDBC we can use Integer variable to hold the No. of Rows affected count & ResultSet Object to hold the DB Results

**java.sql.ResultSet**

* It is an object representation of DB Results produced by Select SQL Query
* ResultSet consists of N number of Rows with each row containing N number of Columns
* Number of rows and columns in Resultset directly depends on the Select SQL query
* ResultSet object is produced by invoking executeQuery() method on JDBC Statements object
* ResultSet object may consist of zero/more OR zero/one rows of data depending on the select SQL query
* If ResultSet consist of zero/more rows of data then **we must use while loop**
* If ResultSet consist of zero/one row of data then **we can use either while loop or if block** (preferred one is if block)
* Once the ResultSet is produced, data from result set can be extracted as follows

1. Move to desired row by calling necessary ResultSet methods

Ex : next(), last(), etc

1. Retrieve the desired column values using

getXXX(<Position of Column in SQL Query>)

OR

getXXX(<column\_name>)

where XXX = JDBC data type corresponding to DB Table Column data type

**Why we need to Close Necessary JDBC Objects:**

* Connections, JDBC Statements, and ResultSet Objects make use of OS resources such as memory. In case of Connections, further DB server resources are consumed
* It is important to close any JDBC object as soon as their job is done; garbage collection should not be relied upon. Forgetting to close any JDBC objects will bring down the application performance

**Request Students to refresh on the below things**

* **Singleton Class & its object creation**
* **Class.forName() method in Core Java**

**Teach Another Way of loading the Driver Class i.e. Class.forName()**

**Assignment 3:**

* Write a JDBC Program which takes the register number, first name, middle name, last name, guardian first name, guardian middle name, guardian last name, isadmin & password information via command line arguments & insert this data into corresponding tables. (**Use this for TRASACTIONS Example**)

**Transactions:-**

* A transaction is a set of one or more SQL Queries that are executed as a unit. So either all of the statements are executed successfully or none of the statements are executed.
* Transactions **helps us to maintain the data consistency**.
* Following steps are followed to handle transaction in JDBC

1. Begin the transaction by Disabling Auto-Commit Mode ( con.setAutoCommit(**false**) )
2. Issue one or more SQL Commands (generally Update, Insert or Delete SQL Commonds)
3. If no errors occur then commit the transaction ( con.commit() )
4. If errors occur then rollback the transaction ( con.rollback() )

**Note:**

1. Whenever there is a scenario to execute more than one Insert / Update / Delete SQL Queries then **we must make use of transactions**
2. Transactions can also be used with one or more Select SQL Queries but it’s of no use
3. Transactions can also be used with ONLY ONE Insert / Update / Delete SQL Query but it’s of no use
4. If there is more than one catch block, then we have to have the rollback statement in all catch blocks.
5. We can also have rollback in the finally block but this impacts the performance. Since rollback is an expensive operation, we should make use of rollback whenever there is a need.

**Assignment 4:**

* Write a JDBC Program to invoke the “StudentUpsert2” stored procedure by passing all the valid data via command line arguments

**Do the above Assignment & with the help of print the time in the console, show the students on how Callable Statements help us to achieve high performance.**

**i.e. Use Transaction program which will have 3 DB interaction**

**Use the above program which will have only one DB interaction**

**So if 30 users as creating the profile simultaneously then 30 X 3 = 90 DB calls will be made in the first case whereas same thing in the second case it will be 30 X 1 = 30 DB calls**

**Connection Pooling**

* Connection pool is a **collection of java.sql.Connection objects**
* Connection pools promote the reuse of connection objects thereby reducing the number of times the connection objects being created & closed
* **Connection pools significantly improve the performance of web applications** because creating connection objects & closing them is costly operation with respect to time and resources. Tasks such as reading DB Url, Network communication, authentication and memory allocation all contribute to the amount of time and resources it takes to create an connection object
* Connection Pool make use of Object Pool Design Pattern & Singleton Design Pattern

**JDBC Surprise Test 1:**

* Write a JDBC Program which takes the register number & password information via command line arguments & update “students\_otherinfo” table with the input password information against the incoming regno

**JDBC Surprise Test 2:**

* Write a JDBC Program which takes the register number & password information via command line arguments, authenticate it by interacting with “students\_otherinfo” table
* If user provides the invalid credentials then this program should print “Invalid User Name / Password” error information in the console
* If user provides the valid credentials then this program should print “Valid User Name / Password” information in the console