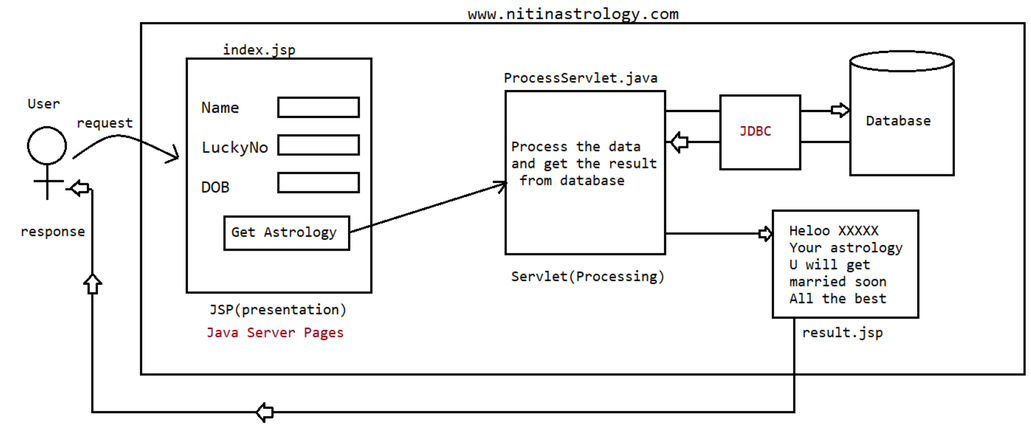
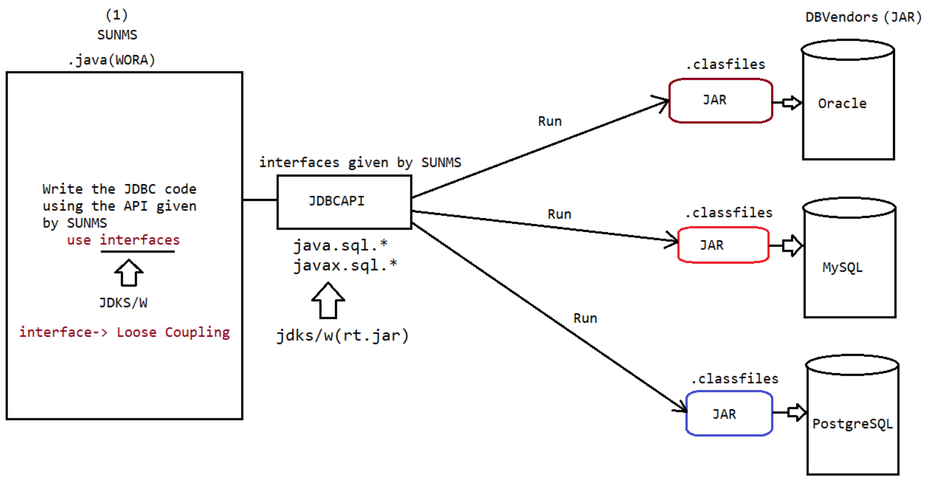


JDBC Architecture



javac,java are the commands executed on commandprompt(os)

⇩

os will search for these commands in an environmental variable called "path".

set path = "location of jdk/bin"

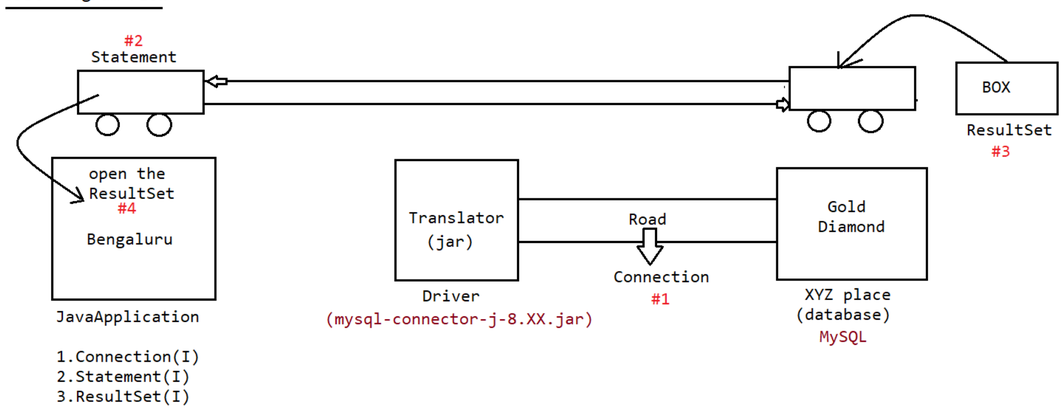
since javac,java commands knows where java.sql packages are present, we need not explicitly tell to javac,java command about the location(rt.jar)

Note:

To inform javac,java command about the location of third party jar used in the project, SunMicroSystem had given a new environmental called "classpath".

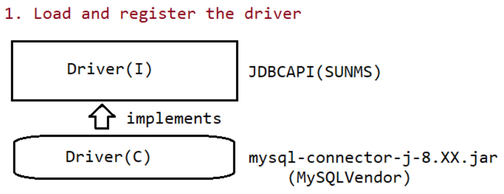
set classpath =";.;location of 3rd party api"

JDBC Program Flow



1.Connection(I)
2.Statement(I)
3.ResultSet(I)

1. Load and register the driver



How to load the .class file explicity into JRE?

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

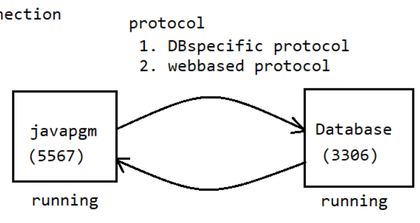
⇧

Driver class is loaded, so in JRE
JDBC environment for MySQL is setup.

```
class Driver
{
   static
   {
        Driver driver = new Driver();
        DriverManager.registerDriver(driver);

   }
}
```

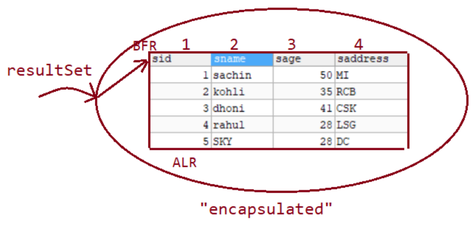
2. Establish the Connection



dbprotocol

protocolName:dbenginename://ipaddress of db :portNoOfDb/dbName

4. Process the resultSet



⇩ takes the cursor to the next row and check whether the row contains data or not if yes returns true otherwise it returns false

```
while (resultSet.next()){

    Integer sid  = resultSet.getInt(1);
    String sname = resultSet.getString(2);
    Integer sage = resultSet.getInt(3);
    String saddr = resultSet.getString(4);

     System.out.println(sid+"\t"+sname+"\t"+sage+"\t"+saddr);
}
```

Wrapping the associated data and its members into single unit is called "Encapsulation".

To access/read the data we need to use "getters".

To set/write the data we need to use "setters".

With core java knowledge we can build standalone applications

Standalone applications
==========
  The applications which are running on single machine are called as "Standalone applications".
      eg: Calculator, MS-word, notepad,....

If we want to develop web-applications then we need to go for "Advanced Java".

What are webapplications?
  The application which provide service over the web/internet are called as "WebApplications".
            eg: gmail.com, facebook.com,ineuron.ai,telusko.com,.......

In java we can develop webapplications using the following technologies
      a. JDBC
      b. Servlet
      c. JSP/Thymeleaf

JDBC(Java Database Connectivity)
      For a Java Application(Normal java class or Servlet) if we want to communicate with database, then we need to go for
      JDBC.
            eg: To get mails information from database
                  To get Astrology information from Database

Servlet
      Whenever some processing logic is required then we need to go for Servlet.
      Servler is meant for Processing logic/Buisness Logic.
            eg: Verify user
                  Communicate with the data
                  Process end user data
JSP
      Whenever presentation logic is required todisplay something to the end user then we need to go for jsp
      Jsp stands for View Component
                  eg: display login page
                        display inbox page
                        display error page
                        display result page

JDBC
  The current version of JDBC is 4.X(4.2V)
      JDK s/w required => jdk8 and above

Steps given by SUNMS to communicate with Database
==========
1. Load and register the Driver.
2. Establish the connection with Database.
3. Create Statement Object and execute the Query.
4. Process the ResultSet.
5. Handle the SQLException if it gets generated.
6. Close the Connection.

1. Load and register the Driver

We need to load and register the Driver as per the DB requirement.
As per the DB specification, we need to set the JRE environment with the DB environment.
Any class of DB vendor, we say at is Driverclass, iff it has implemented a interface called "Driver".

MySQL =======> The implementation class of MySQLjar is "Driver".

2. Establish the Connection with database
  public static Connection getConnection(String, java.util.Properties) throws java.sql.SQLException;
  public static Connection getConnection(String, String, String) throws java.sql.SQLException;
  public static Connection getConnection(String) throws java.sql.SQLException;

Connection connection = DriverManager.getConnection(url,username,password);
The above line would create connection Object,but Connection is an interface,can object to interface be created?
Answer: No, for an interface instantiation is not possible.
In this line, an Object to a class which implements an interface called Connection is created and we hold the
the reference of the object with the interface name.
This is done to promote loose coupling in java.
The above code would also represent "abstraction".

3.Create Statement Object and execute the Query.
Statement statement = connection.createStatement();
The above line would create statement Object,but Statement is an interface,can object to interface be created?
Answer: No, for an interface instantiation is not possible.
In this line, an Object to a class which implements an interface called Statement is created and we hold the
the reference of the object with the interface name.
This is done to promote loose coupling in java.
The above code would also represent "abstraction".

ResultSet resultSet =statement.executeQuery(sqlSelectQuery);
The above line would create statement Object,but ResultSet is an interface,can object to interface be created?
Answer: No, for an interface instantiation is not possible.
In this line, an Object to a class which implements an interface called ResultSet is created and we hold the
the reference of the object with the interface name.
This is done to promote loose coupling in java.
The above code would also represent "abstraction".

jdbc pgm to retrieve all records from database
==========

```
import java.sql.*;

class TestApp {
      public static void main(String[] args) {

            Connection connection  = null;
            Statement  statement   = null;
            ResultSet  resultSet   = null;

            try{
```

```
                //Step1. Load and register the Driver
                Class.forName("com.mysql.cj.jdbc.Driver");
                System.out.println("Driver loaded succesfully....");

                //Step2. Establish the Connection with database
                String url = "jdbc:mysql://localhost:3306/octbatch";

                //username and password would vary from user to user
                String userName = "root";
                String passWord = "root123";

                connection = DriverManager.getConnection(url,userName,passWord);
                System.out.println("connection established succesfully...");
                System.out.println("The implement class name is "+
                      connection.getClass().getName());


                //Step3. Create statement Object and send the query
                String sqlSelectQuery = "select sid,sname,sage,saddress from
student";
                statement = connection.createStatement();
                System.out.println("The implementation class name
is ::"+statement.getClass().getName());

                resultSet =statement.executeQuery(sqlSelectQuery);
                System.out.println("The implementation class name
is ::"+resultSet.getClass().getName());

                System.out.println();

                System.out.println("SID\tSNAME\tSAGE\tSADDRESS");

                //Step4. Process the resultSet
                while (resultSet.next())
                {
                      Integer sid  = resultSet.getInt(1);
                      String sname = resultSet.getString(2);
                      Integer sage = resultSet.getInt(3);
                      String saddr = resultSet.getString(4);

                      System.out.println(sid+"\t"+sname+"\t"+sage+"\t"+saddr);
                }
          }catch (ClassNotFoundException ce){
                ce.printStackTrace();
          }catch(SQLException se){
                se.printStackTrace();
          }catch(Exception e){
                e.printStackTrace();
          }finally{

                //closing the resources
                if (connection!=null)
                {
                      try
                      {
                            connection.close();
                            System.out.println("Connection closed...");
                      }
                      catch (SQLException se){
```

```
                            se.printStackTrace();
                        }
                    }
                }

        }
}
```

output

```
D:\jdbcoctbatch>echo %path%
C:\Program Files\Java\jdk1.8.0_202\bin

D:\jdbcoctbatch>echo %classpath%
;.;D:\jars\mysql-connector-j-8.0.31.jar

D:\jdbcoctbatch>javac TestApp.java

D:\jdbcoctbatch>java TestApp
Driver loaded succesfully....
connection established succesfully...
The implement class name is com.mysql.cj.jdbc.ConnectionImpl
The implementation class name is ::com.mysql.cj.jdbc.StatementImpl
The implementation class name is ::com.mysql.cj.jdbc.result.ResultSetImpl

SID     SNAME   SAGE    SADDRESS
1       sachin  50      MI
2       kohli   35      RCB
3       dhoni   41      CSK
4       rahul   28      LSG
5       SKY     28      DC

Connection closed...
```