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
==========

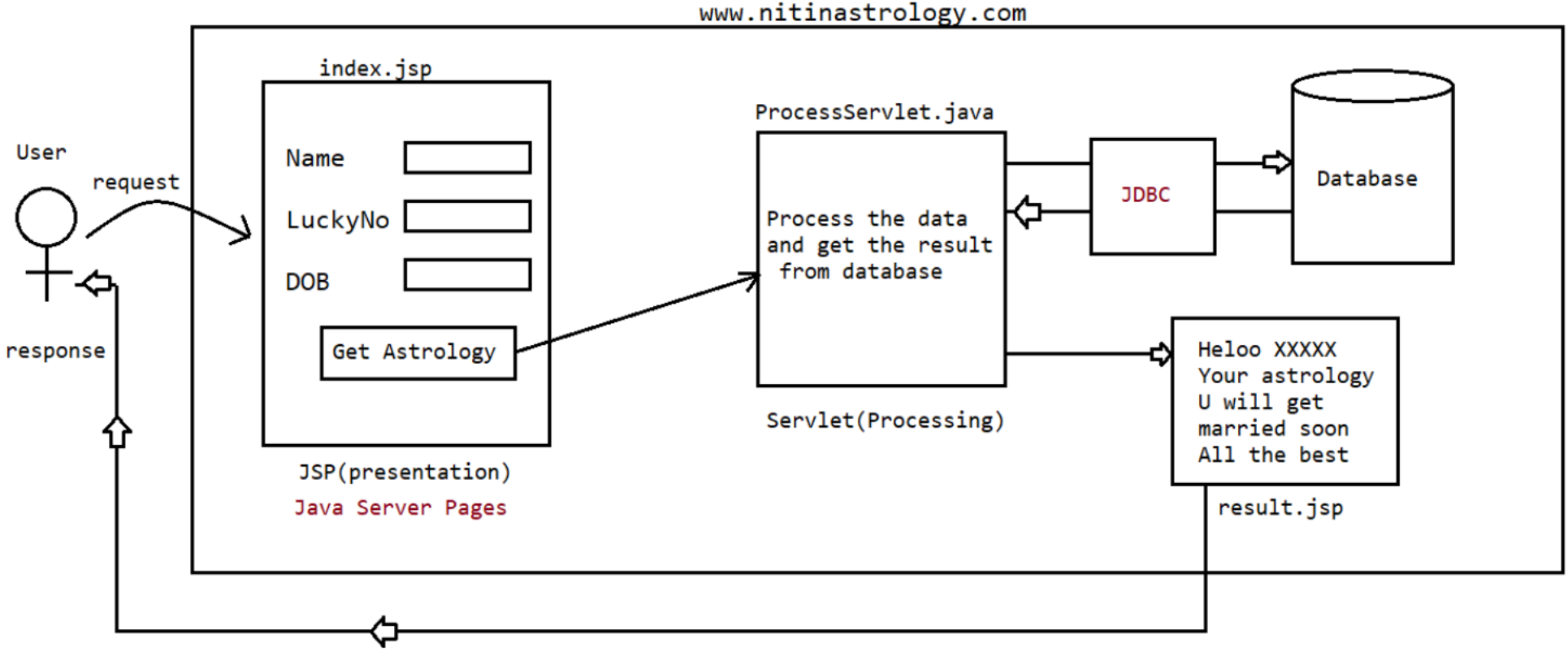
```
import java.sql.*;

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){
```

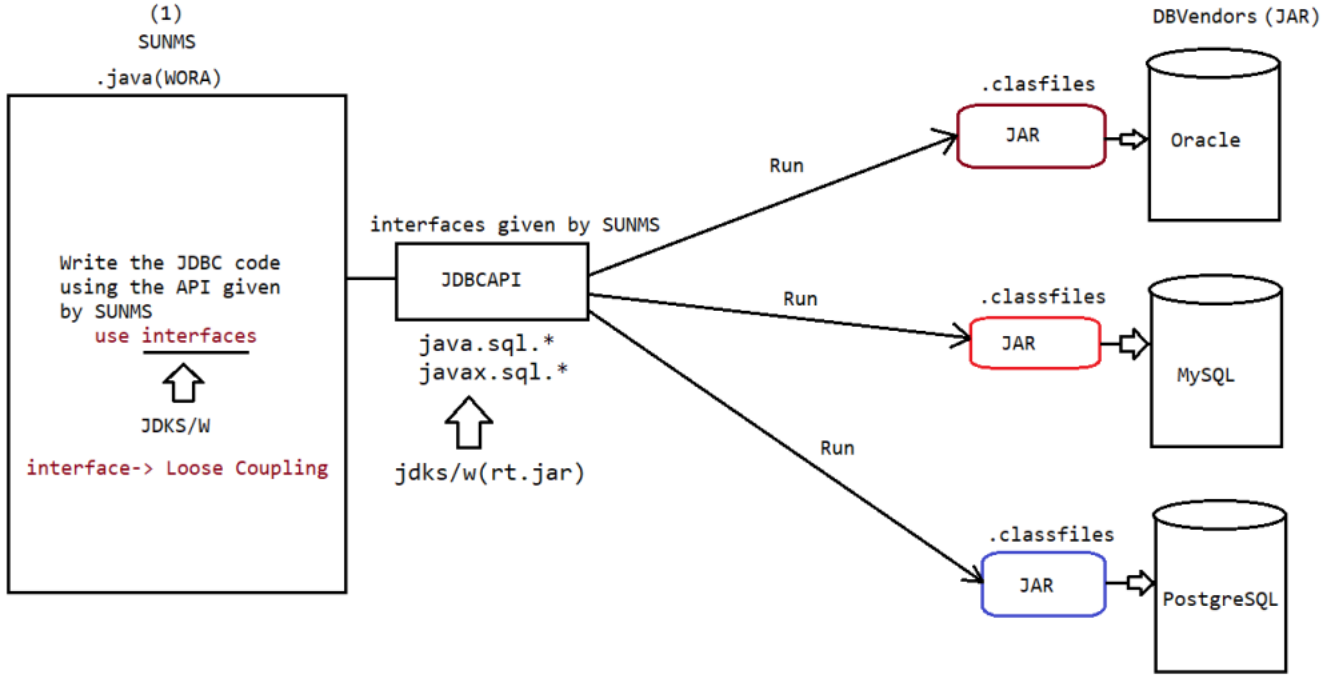
```
                            se.printStackTrace();
                        }
                    }
                }

        }
}
```

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...
```


www.nitinastrology.com
index.jsp
Name
LuckyNo
DOB
Get Astrology
JSP(presentation)
Java Server Pages
User
request
response
ProcessServlet.java
Process the data
and get the result
from database
Servlet(Processing)
JDBC
Database
Heloo XXXXX
Your astrology
U will get
married soon
All the best
result.jsp

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"
```

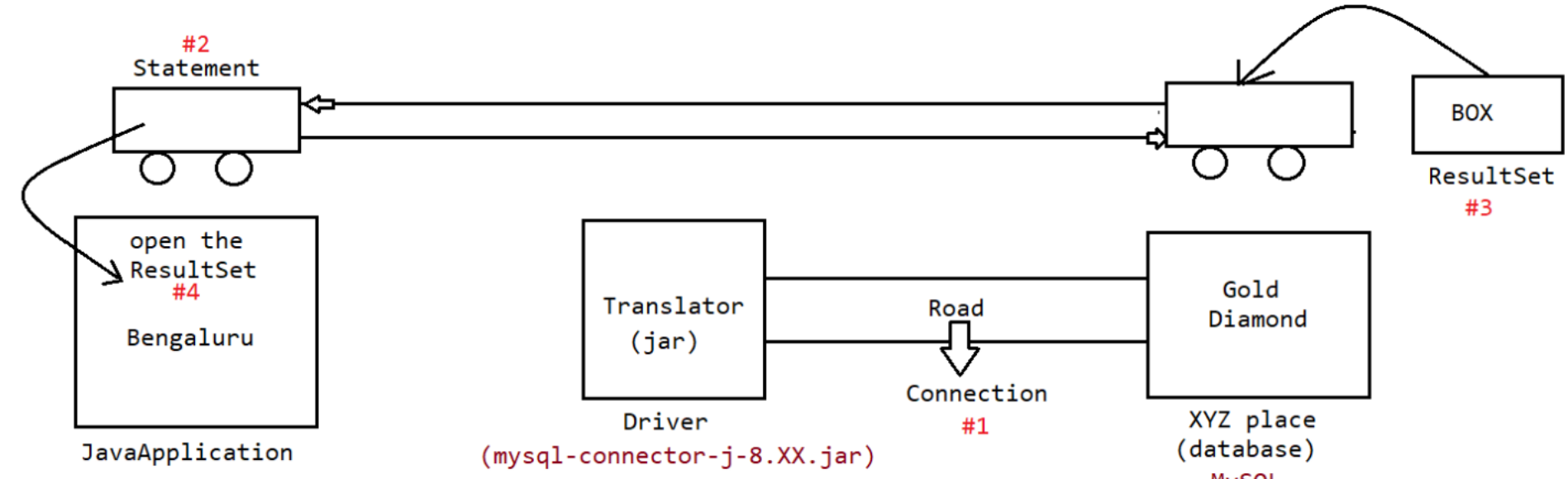
since javac,java commands knows where java.sql packages are present, we need not explicitly tell to javac,java command about the location(rt.jar)

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".

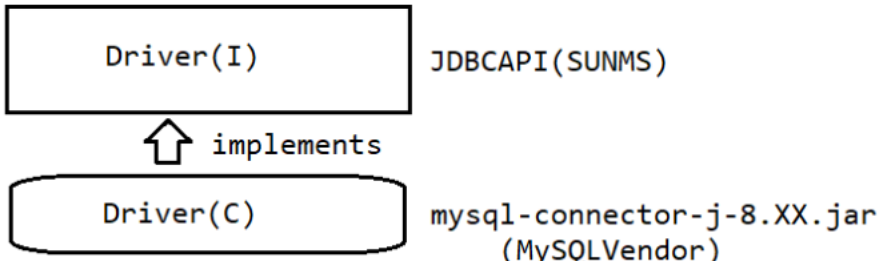
```
set classpath =";.;location of 3rd party api"
```

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?

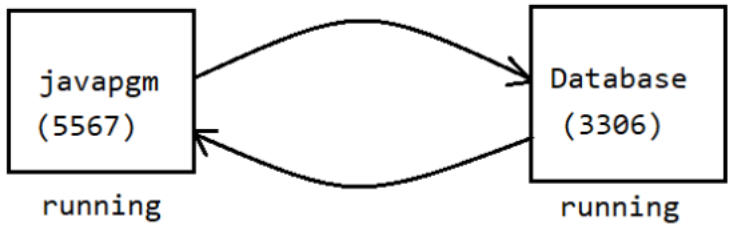
```
Class.forName("com.mysql.cj.jdbc.Driver");
```

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

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

   }
}
```

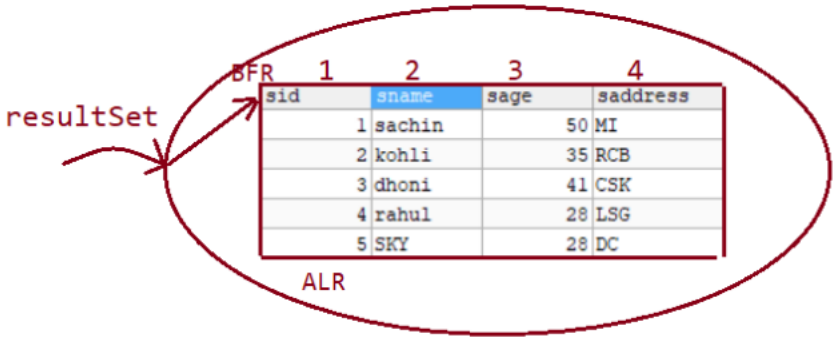
2. Establish the Connection

protocol
1. DBspecific protocol
2. webbased protocol



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);
}
```

"encapsulated"

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".

From JDBC4.X version onwards,there is a facility of "autoloading".

Q> What is autoloading in JDBC?
      Loading and register the driver is done automatically,based on the url supplied by the user.
      Behind the scenes
            a. check the url
            b. based on the url supplied, go to classpath environmental variable
            c. open the relevant jar
            d. go to META-INF/services folder
            e. open java.sql.Driver file
            f. read the file and load the class supplied in the file

Note:
    Using resultSet object, we can retrieve the records based on the column names also.
    If java pgm and database engine is running in the same program with the default port no for database then
    url can be of the following type
            String url = "jdbc:mysql:///octbatch".

Program to demonstrate select operation using JDBC
==========
```
package in.ineuron.main;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

//JDBC4.X autoloading feature is enabled.
public class SelectApp {

      public static void main(String[] args) throws ClassNotFoundException, SQLException {

            // Step2. Establish the Connection
            String url = "jdbc:mysql:///octbatch";
            String user = "root";
            String password = "root123";
            Connection connection = DriverManager.getConnection(url, user, password);
            System.out.println("CONNECTION object created...");

            // Step3. Create statement Object and send the Query
            Statement statement = connection.createStatement();
            System.out.println("STATEMENT object created...");

            // Step4. Execute the Query and Process the resultSet
            String sqlSelectQuery = "select sid,sname,sage,saddress from student";
            ResultSet resultSet = statement.executeQuery(sqlSelectQuery);
            System.out.println("RESULTSET object created...");
            System.out.println("SID\tSNAME\tSAGE\tSADDRESS");
            while (resultSet.next()) {

                  int sid = resultSet.getInt("sid");
                  String sname = resultSet.getString("sname");
```

```
                int sage = resultSet.getInt("sage");
                String saddress = resultSet.getString("saddress");
                System.out.println(sid + "\t" + sname + "\t" + sage + "\t" +
saddress);
            }

            // Step6. Close the resources
            resultSet.close();
            statement.close();
            connection.close();
            System.out.println("Closing the resources...");

      }
}
Output
Loading the driver...
CONNECTION object created...
STATEMENT object created...
RESULTSET object created...
SID   SNAME SAGE  SADDRESS
1     sachin            50                MI
2     kohli       35                RCB
3     dhoni       41                CSK
4     rahul       28                LSG
5     SKY               28                DC
Closing the resources...
```

Note
According to DBA specification, all SQL commands are categorised into following types

a. DDL(Data Definition Language)
   eg: Create table, alter table, drop table, etc...
b. DML(Data Manipulation Language)
   eg: insert,update,delete
c. DQL(Data Query Language)
   eg: Select
d. DCL(Data Control Language)
   eg: Alter password, grant access,....
e. DA command(Database Administrator commands)
   eg: start audit
       stop audit
f. TCL(Transaction Control Language)
   eg: commit, rollback,savepoint,....

According to Java Developer point of view, all SQL operations are classified into 2 types

a. Select operation(DQL)
b. Non-Select Operation(DML,DDL,...)

Through Statement Object we need to execute the Query and to exectue the Query we need to make a call
to a method as shown below.

a. executeQuery()
b. executeUpdate()
c. execute()

a. executeQuery()
   This method is used only if we perform select operation.

```
      Because of  this method execution, we will get a group of records which are
represented as "ResutlSet" object.
            public ResultSet executeQuery(String sqlSelectQuery) throws
SQLException;
                  eg: ResultSet resultSet =statement.executeQuery("select
sid,sname,sage,saddress from student");

b. executeUpdate()
      This method is used for "Non-Select Operations" like(Insert|Update|Delete)
      Because of this method execution, we won't get group of records, we will get
a numeric value which represents
      the number of rows affected. So return type of the method is "int".
                  public int executeUpdate(String sqlNonSelectQuery) throws
SQLException;
                  eg: int rowAffected = statement.executeUpdate("delete from
student where sid = 10");
                        System.out.println("No of rows affected is ::
"+rowAffected);

c. execute()
            we can use this method for both select and nonselect operation
            if we don't know the type of query at the begining and if is available
dynamically at the runtime then we need to use
            this method for execution.
                  public boolean execute(String sql) throws SQLException;
                  eg: boolean value = statement.execute(dynamicQuery);
                        if(value == true)
                        {
                                    //select Query
                                    ResultSet resultSet = statement.getResultSet();

                                    //process the resultSet
                        }
                        else
                        {
                                    //nonSelect Query
                                    int rowCount = statement.getUpdateCount();
                                    System.out.println("Number of rows affected
is :: "+rowCount);
                        }

Formatting SQlQueries using dynamic input
==================================

1st approach
==========
sname = scanner.next();
sage =scanner.nextInt();
saddress = scanner.next();
String sqlInsertQuery = "insert into
student(`sname`,`sage`,`saddress`)values('"+sname+"',"+sage+",'"+saddress+"')";

2nd approach
===========
sname = scanner.next();
sage =scanner.nextInt();
saddress = scanner.next();
sname = "'"+sname+"'";
saddress = "'"+saddress+"'";
```

```
String sqlInsertQuery = "insert into
student(`sname`,`sage`,`saddress`)values("+sname+","+sage+","+saddress+")";

3rd approach
===========
      The above 2 approaches are not recomended, to do formatting we prefer using
String class format() as shown below.
            public static  String format(String format, Object... args)

sname = scanner.next();
sage =scanner.nextInt();
saddress = scanner.next();
String sqlInsertQuery =String.format("insert into
student(`sname`,`sage`,`saddress`) values ('%s',%d,'%s')",
                                                              sname,sage,saddress);

Write a  JDBC code to take dynamic input from the user to perform insert operation
package in.ineuron.main;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Scanner;

//JDBC4.X autoloading feature is enabled.
public class InsertApp {

      public static void main(String[] args) throws SQLException {

            // Step2. Establish the Connection
            String url = "jdbc:mysql:///octbatch";
            String user = "root";
            String password = "root123";
            Connection connection = DriverManager.getConnection(url, user,
password);
            System.out.println("connection object created...");

            // Step3. Create statement Object and send the Query
            Statement statement = connection.createStatement();
            System.out.println("statement object created...");

            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter the name of the student :: ");
            String sname = scanner.next();

            System.out.print("Enter the age of the student :: ");
            int sage = scanner.nextInt();

            System.out.print("Enter the address of the student :: ");
            String address = scanner.next();

            System.out.print("Enter the gender of a student:: ");
            String gender = scanner.next();

            // Step4. Execute the Query and Process the resultSet
            String sqlInsertQuery = String.format("insert into
student(`sname`,`sage`,`saddress`,`sgender`) values ('%s',%d,'%s','%s')",
```

```
                  sname, sage, address,gender);

            System.out.println(sqlInsertQuery);

            int rowAffected = statement.executeUpdate(sqlInsertQuery);
            System.out.println("No of rows affected is :: " + rowAffected);

            // Step6. Close the resources
            statement.close();
            connection.close();
            scanner.close();
            System.out.println("closing the resources...");

      }
}
Output
connection object created...
statement object created...
Enter the name of the student :: mandana
Enter the age of the student :: 27
Enter the address of the student :: kodagu
Enter the gender of a student:: F
insert into student(`sname`,`sage`,`saddress`,`sgender`) values
('mandana',27,'kodagu','F')
No of rows affected is :: 1
closing the resources...


Note:
 While writing JDBC code, the following steps are common
            a. Establishing the connection
            b. closing the resources
            c. handling the exception
 What would vary in every code is
            a. query will be different
                  1. if it select process the ResultSet object.
                  2. if it non-select, process the integer value.

For the above 3 steps, the common util code is as shown below
package in.ineuron.util;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

public class JdbcUtil {

      private JdbcUtil() {
      }

      static {
            //Step1: loading and register the Driver
            try {
                  Class.forName("com.mysql.cj.jdbc.Driver");
            } catch (ClassNotFoundException ce) {
                  ce.printStackTrace();
```

```
			}
		}

		public static Connection getJdbcConnection() throws SQLException {
			// Step2. Establish the Connection
			String url = "jdbc:mysql:///octbatch";
			String user = "root";
			String password = "root123";
			Connection connection = DriverManager.getConnection(url, user,
password);
			System.out.println("connection object created...");
			return connection;
		}
		public static void cleanUp(Connection con, Statement statement, ResultSet
resultSet) throws SQLException {
			// Step6. Close the resources
			if (con != null) {
				con.close();
			}
			if (statement != null) {
				statement.close();
			}
			if (resultSet != null) {
				resultSet.close();
			}

		}
}
```

Note:
  In the above code, the url pattern, username and password is hardcode.
  These values would vary from user to user.
  To set these values, we use properties file approach as shown below.

```
package in.ineuron.util;

import java.io.FileInputStream;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Properties;

public class JdbcUtil {

		private JdbcUtil() {
		}

		static {
			// Step1: loading and register the Driver
			try {
				Class.forName("com.mysql.cj.jdbc.Driver");
			} catch (ClassNotFoundException ce) {
				ce.printStackTrace();
			}
		}
```

```
	public static Connection getJdbcConnection() throws SQLException, IOException
{

		// Take the data from properties file
		FileInputStream fis = new FileInputStream("D:\\octbatchjdbcpgm\\
JDBCStandardApp\\application.properties");
		Properties properties = new Properties();
		properties.load(fis);

		// Step2. Establish the Connection
		Connection connection =
DriverManager.getConnection(properties.getProperty("url"),
				properties.getProperty("username"),
properties.getProperty("password"));
		System.out.println("connection object created...");
		return connection;
	}

	public static void cleanUp(Connection con, Statement statement, ResultSet
resultSet) throws SQLException {
		// Step6. Close the resources
		if (con != null) {
			con.close();
		}

		if (statement != null) {
			statement.close();
		}
		if (resultSet != null) {
			resultSet.close();
		}

	}
}
application.properties
==================
url=jdbc:mysql:///octbatch
username=root
password=root123
```

Properties
========
 It represents the data in the form of K,V pair
 It represents an object of type java.util.Properties
 Properties object holds the data which would be changing frequently in our application.

Program to demonstrate the usage of Properties object in java
==
```
import java.io.*;
import java.util.*;
class PropertiesApp
{
      public static void main(String[] args)throws Exception
      {
            FileInputStream fis = new FileInputStream("application.properties");
            Properties properties = new Properties();
            properties.load(fis);

            String url  = properties.getProperty("url");
            String user = properties.getProperty("user");
            String password = properties.getProperty("password");

            System.out.println("URL  IS ::"+url);
            System.out.println("USER IS ::"+user);
            System.out.println("PWD  IS ::"+password);

      }
}
```
application.properties
==================
```
url =jdbc:oracle:thin:@localhost:1521:XE
user=System
password=root123
```

Output
```
D:\jdbcoctbatch>javac PropertiesApp.java

D:\jdbcoctbatch>java PropertiesApp
URL  IS ::jdbc:mysql:///octbatch
USER IS ::root
PWD  IS ::root123

D:\jdbcoctbatch>java PropertiesApp
URL  IS ::jdbc:oracle:thin:@localhost:1521:XE
USER IS ::System
PWD  IS ::root123
```

PrepardStatement(I)
===============
 public abstract PreparedStatement prepareStatement(String query) throws SQLException;

While using PreparedStatement the query woudl be as shown below
```
      String sqlInsertQuery ="insert into
student(`sname`,`sage`,`saddress`,`sgender`)values(?,?,?,?)";
      PreparedStatement pstmt = con.prepareStatement(sqlInsertQuery);
```

```
      //query compiled ready for execution
      pstmt.setString(1,"apeksha");
      pstmt.setInt(2,26);
      pstmt.setString(3,"MI");
      pstmt.setString(4,"F");

      int rowCount = pstmt.executeUpdate();

      refer: JDBCPreparedStatementApp

JDBC Project Requirement
====================
      1. Develop a console based application in the following manner
            a. Create a menu for the user to perform CRUD operation as shown below
                        1. Press 1 for Insert operation
                        2. Press 2 for select operation
                        3. Press 3 for Update operation
                        4. Press 4 for Delete operation
                        5. Press 5 for exit
                              Note: anything above 5 tell invalid operation

            b. If user presses 1
                  a. Take inputs from the user to accept the data like
                              1. sid(pk)  2. sname 3. sage 4. saddress
                  b. perform insertion operation
                  c. display suitable message as
                              a. record inserted succesfully
                              b. record insertion failed

            c. If user presses 2
                  a. Take inputs from the user to accept the data like
                              1. sid(pk)
                  b. perform select operation
                  c. display suitable message as
                              a. display the details in table format.
                              b. record not available for the given id.

            d. b. If user presses 3
                  a. Take inputs from the user to accept the data like
                              1. sid(pk)
                  b. for the entered id display the details first
                                    sid        : XXXXX (no change becoz it is pk)
                                    sname : XXXXX   enter new sname :: XXXXX
                                    sage     : XXXXX   enter new sage     :: YYYYY
                                    saddr   : XXXXX    enter new saddr   :: YYYYY
                  c. display suitable message as
                              a. record updated succesfully
                              b. record updation failed

            e. If user presses 4
                  a. Take inputs from the user to accept the data like
                              1. sid(pk)
                  b. perform delete operation
                  c. display suitable message as
                              a. record deleted succesfully.
                              b. record not available for the given id.

Advantages of PreparedStatement
===============================
```

1. Performance is very high compared to Statement approach becoz query will be compiled only once.
2. Since we don't send the query multiple times b/w java application and database traffic will be reduced.
3. inputs to the query need not be supplied at the begining dynamically we can supply the inputs.
4. inputs to the query can be supplied just like java style, no need to perform formatting as per the DB specification.
5. Best suitable for inserting Date values.
6. Best sutiable for insertion of BLOB's and CLOB's (image and pdf files).
7. It prevents SQLInjection Attack.

Limitation of PreparedStatement
==========

```
            Statement stmt = connection.createStatement();//[samsung phone ----> airtelsim,jiosim,vi]GSMA
                  stmt.executeUpdate("insert into student values ()...");
                  stmt.executeQuery("select * from student");
                  stmt.executeUpdate("delete from student wher....");
```

      One statement object can be used to execute mulitple query but with no change in inputs.

```
      PreparedStatement pstmt = connection.prepareStatement("select * from student");//[Jio Phone----> jio sim]CDMA
                  pstmt.executeQuery();
```

      One PreparedStatement object is restricted to only one query, that query can be executed multiple times
      with change in input.

Which of the following represent valid statements ?
   1. delete from employee where ename = ?(valid)
   2. delete from employee ? ename = ? (invalid)
   3. delete from ? where ename = ? (invalid)
   4. delete ? employees where ename = ?(invalid)

Note: we can use ? only in the place of input values and we cannot use in the place of sql keywords, tablenames
          and column names.

Static query vs Dynamic query
==========

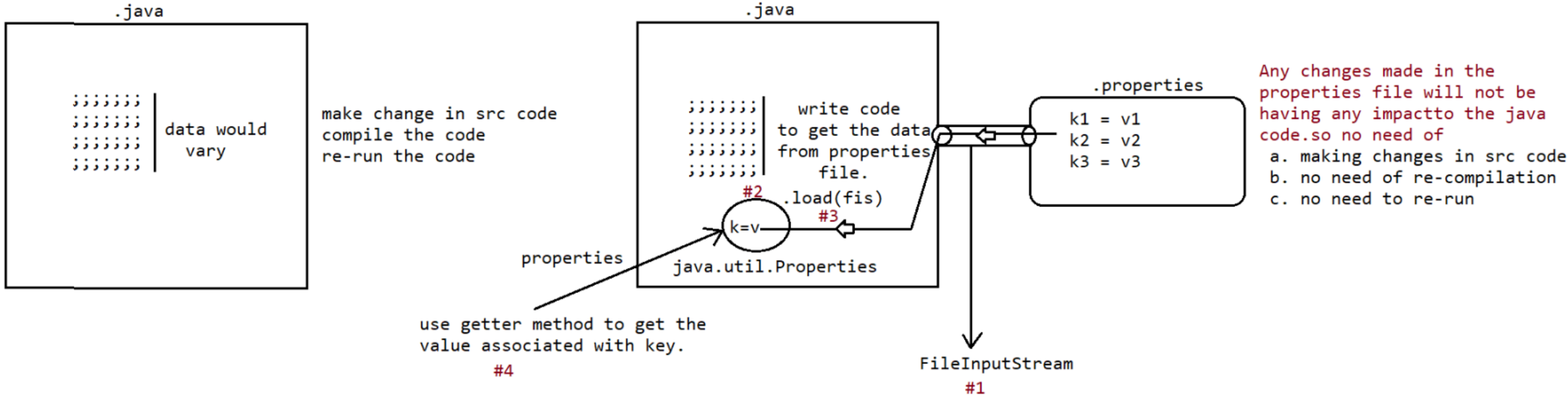
The sql query without positional parameter(?) is called static query.

```
            eg: delete from employee where ename = 'sachin';
```

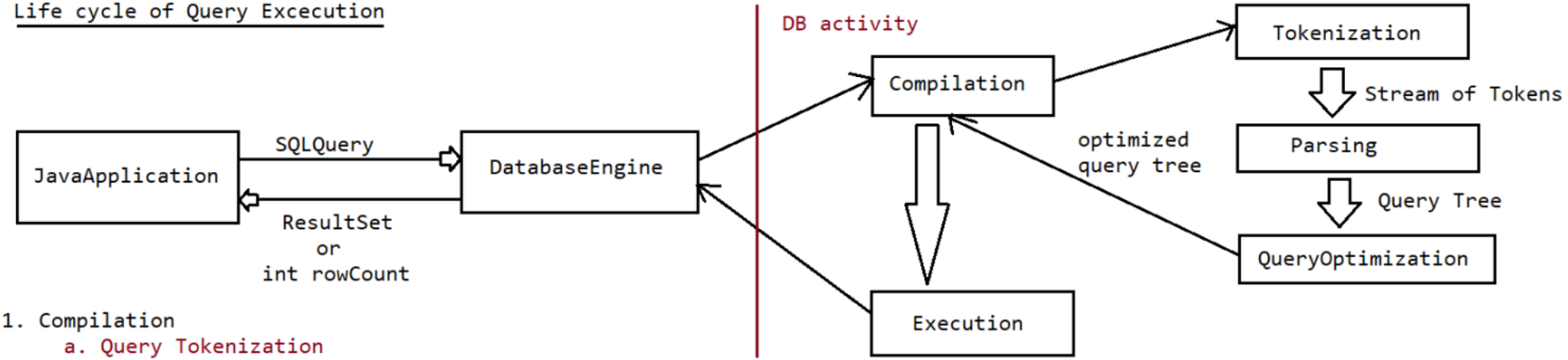
The sql query with positional parameter(?) is called dynamic query

```
            eg: delete from employee where ename = ?
                  select eid,ename,esal from employee where esal > ?
```

Note:
Simple Statement object can be used for static queries, where as Preparedstatement object can be used for static queries
and dynamic queries also.


.java
;;;;;;;
;;;;;;;
;;;;;;;
;;;;;;;
data would
vary
make change in src code
compile the code
re-run the code
.java
;;;;;;;
;;;;;;;
;;;;;;;
;;;;;;;
write code
to get the data
from properties
file.
#2
.load(fis)
#3
k=v
properties
java.util.Properties
use getter method to get the
value associated with key.
#4
FileInputStream
#1
.properties
k1 = v1
k2 = v2
k3 = v3
Any changes made in the
properties file will not be
having any impactto the java
code.so no need of
a. making changes in src code
b. no need of re-compilation
c. no need to re-run

Life cycle of Query Excecution



1. Compilation
   a. Query Tokenization
   b. Query Parsing
   c. Query Optimization

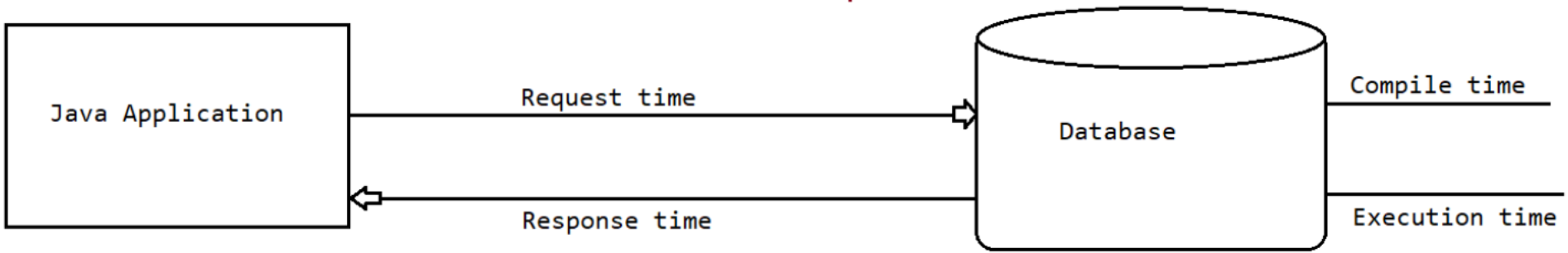
2. Execution
   once compilation is successful the optimized query is taken as input and the query will be executed by the query intrepreter.

3. Fetch the result
   Result will be sent in the form of ResultSet(selectquery) or in the form of rowcount(nonselect query) to the java applicaton by the database engine.

Need of PreparedStatement

```
Statement stmt = con.createStatement();
ResultSet rs = stmt.executeQuery("select * from employees");
```

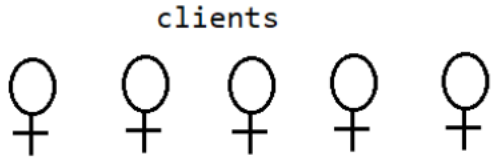
⇦ one statement object can be used to run multiple queries.




```
Total time per query = Req.Time + C.T + E.T + Resp.T
                     =   1ms    + 1ms + 1ms + 1ms
                     = 4ms

   For 1000 queries = 1000*4ms
                    = 4000ms
```

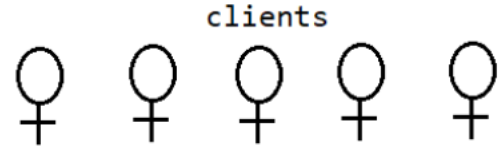
IRCTC

```
select * from trains where source = 'XXXX' and destination = 'XXXX'
```



BMS

```
select * from theatres where city = 'XXXX' and movie ='XXXX';
```



In the above context, same query is executed mulitple times with change in the input.

If we use Statement object approach

a. same query has to be compiled every time for every client request, this would create performance issue.

To resolve this problem we need to use "PreparedStatement(I)".

Advantage of using PreparedStatement

a. Query will be compiled only once even though we are executing in mulitple time with different inputs.
b. through this performance is increased.

```
PreparedStatement prepareStatement(String query) throws SQLException;
```

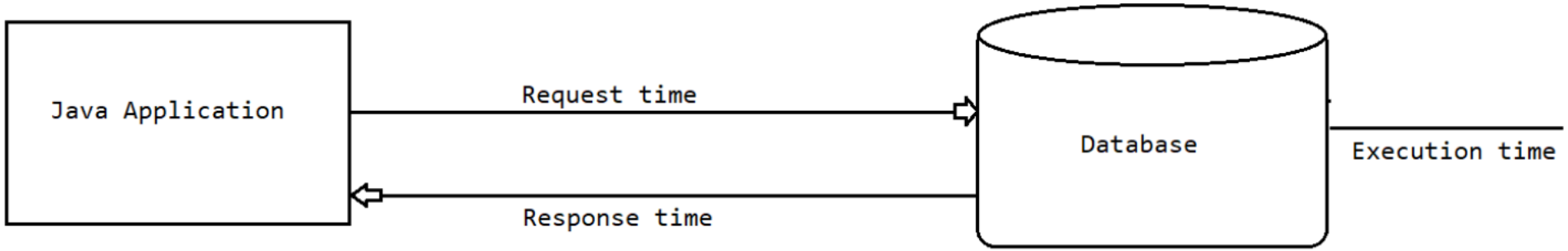
```
PreparedStatement pstmt = con.prepareStatement(sqlQuery)
```

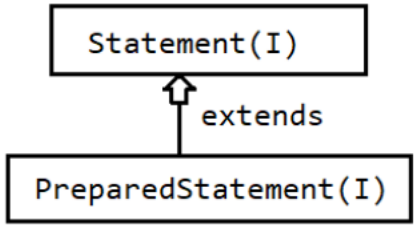
1. At this line query will be sent to database engine.
2. DB engine will compile the query and stores in the database.
3. The preCompiled query will be sent to the java application in the form of PreparedStatement Object.

Note: PreparedStatement is also called as "PreCompiledQuery".

Note: PreparedStatement is also called as "PreCompiledQuery".

```
pstmt.setXXXX(object,object);
pstmt.executeQuery()
pstmt.executeUpdate()
```






```
Total time per query = Req.Time +  E.T + Resp.T
                     =    1ms    +  1ms + 1ms
                     =    3ms

   For 1000 queries = 1000* 3
                    = 3000ms
```

place holders/positional paremeters

1 2 3 4

```
                                                                    1 2 3 4
String sqlInsertQuery ="insert into student(`sname`,`sage`,`saddress`,`sgender`)values(?,?,?,?)";
PreparedStatement pstmt = con.prepareStatement(sqlInsertQuery);⇦Query is compiled at DB level

   //query compiled ready for execution
    pstmt.setString(1,"apeksha");
    pstmt.setInt(2,26);                  using pstmt can we execute only one query
    pstmt.setString(3,"MI");             multiple times with change in the inputs.
    pstmt.setString(4,"F");

    int rowCount = pstmt.executeUpdate();
```

Handling Date values for Database operations
====================================
=> Sometimes as the part of programing requirement,we have to insert and retrieve Date like
   DOB,DOJ,DOM,DOP...wrt database.
=> It is not recommended to maintain date values in the form of String,b'z comparisons will become
     difficult.

In Java we have two Date classes
 1. java.util.Date
 2. java.sql.Date

=> java.sql.Date is the child class of java.util.Date.

=> java.sql.Date is specially designed class for handling Date values wrt database.
     Otherthan database operations,if we want to represent Date in our java program then we should
     go for java.util.Date.
=> java.util.Date can represent both Date and Time where as java.sql.Date represents only Date but
     not time.

```
1) class Test
2) {
3)    public static void main(String[] args)
4)    {
5)          java.util.Date udate=new java.util.Date();
6)          System.out.println("util Date:"+udate);
7)          long l =udate.getTime();
8)          java.sql.Date sdate= new java.sql.Date(l);
9)          System.out.println("sql Date:"+sdate);
10)    }
11) }
```

```
util Date:Mon Mar 20 19:07:29 IST 2017
sql Date:2017-03-20
```

Differences between java.util.Date and java.sql.Date
    java.util.Date
1) It is general Utility Class to handle Dates in our Java Program.
2) It represents both Data and Time.

   java.sql.Date
1) It is specially designed Class to handle Datesw.r.t DB Operations.
2) It represents only Date but not Time.

Note: In sql package Time class is availble to represent Time values and TimeStamp class is
                available to represent both Date and Time.

-> Inserting Date Values into Database:
        Various databases follow various styles to represent Date.
   Eg:
    Oracle: dd-MMM-yy eg: 28-May-90
    MySQL : yyyy-mm-dd eg: 1990-05-28

=> If we use simple Statement object to insert Date values then we should provide Date value in   the database supported format,which is difficult to the

programmer.
=> If we use PreparedStatement,then we are not required to worry about database supported form,
   just we have to call
       pst.setDate (2, java.sql.Date);
This method internally converts date value into the database supported format.
Hence it is highly recommended to use PreparedStatement to insert Date values into database.

Steps to insert Date value into Database:
 => DB: create table users(name varchar2(10),dob date);
1. Read Date from the end user(in String form)
       System.out.println("Enter DOP(dd-mm-yyyy):");
       String dop=sc.next();

2. Convert date from String form to java.util.Date form by using SimpleDateFormat object.
       SDF sdf= new SDF("dd-MM-yyyy");
       java.util.Date udate=sdf.parse(dop);

3. convert date from java.util.Date to java.sql.Date
       long l = udate.getTime();
       java.sql.Date sdate=new java.sql.Date(l);

4. set sdate to query
       pst.setDate(2,sdate);

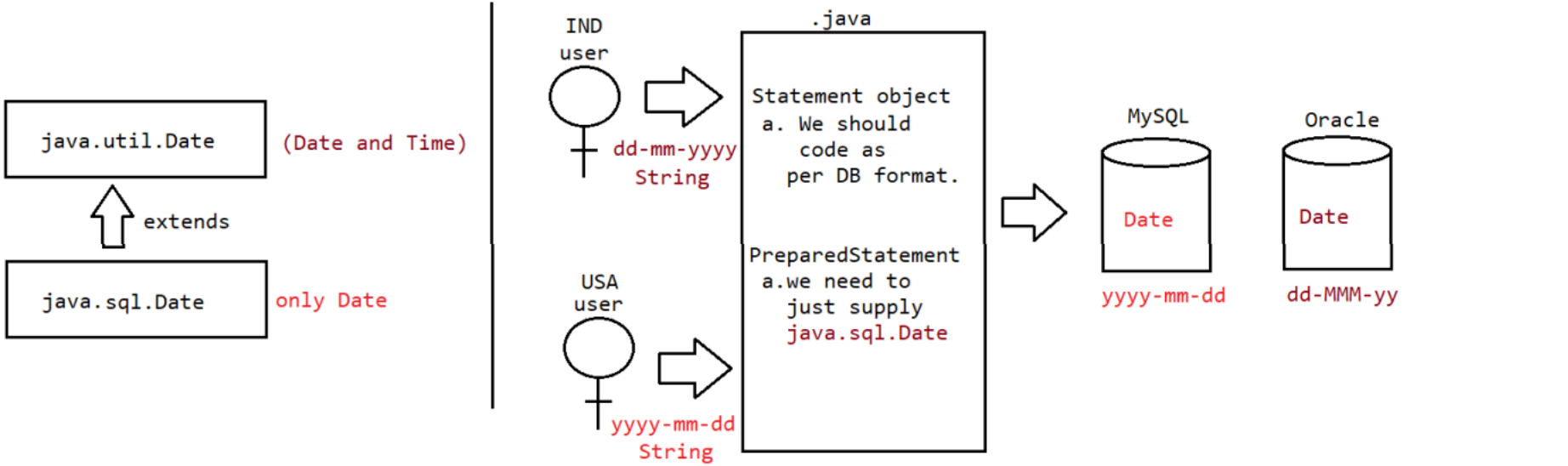
5. int rowAffected= pst.executeUpdate();//Execute the query.

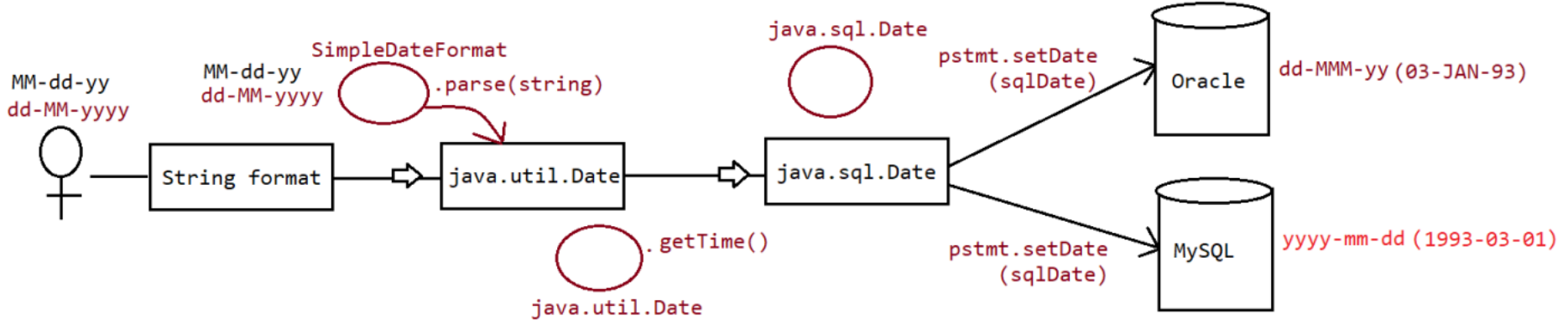
***Note:
If end user provides Date in the form of "yyyy-MM-dd" then we can convert directly that String into
 java.sql.Date form as follows...
 eg:: String s = "1980-05-27";
         java.sql.Date sdate=java.sql.Date.valueOf(s);

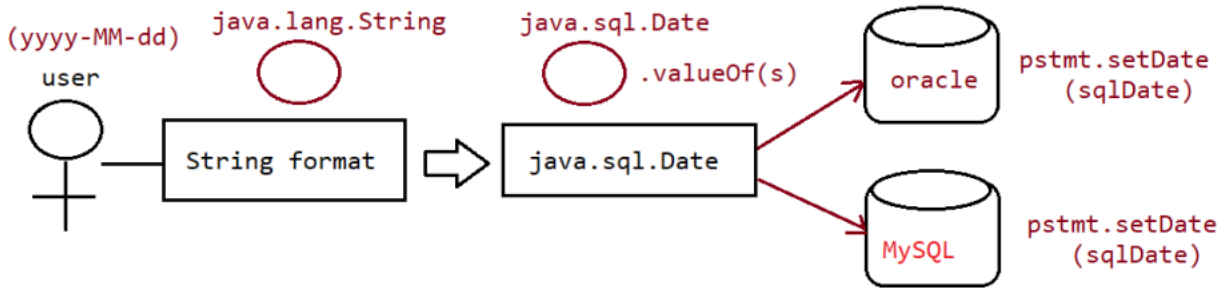
Retrieving Date value from the database
=======================================
=> For this we can use either simple Statement or PreparedStatement.
=> The retrieved Date values are Stored in ResultSet in the form of "java.sql.Date" and we can get
      this value by using getDate() method.
=> Once we got java.sql.Date object,we can format into our required form by using
      SimpleDateFormat object.

Sequence
========
1. Database
       (java.sql.Date)sqldate = rs.getDate(2);
2. Our required String Form
       String s = sdf.format(sqldate);
3. String s holds the date.

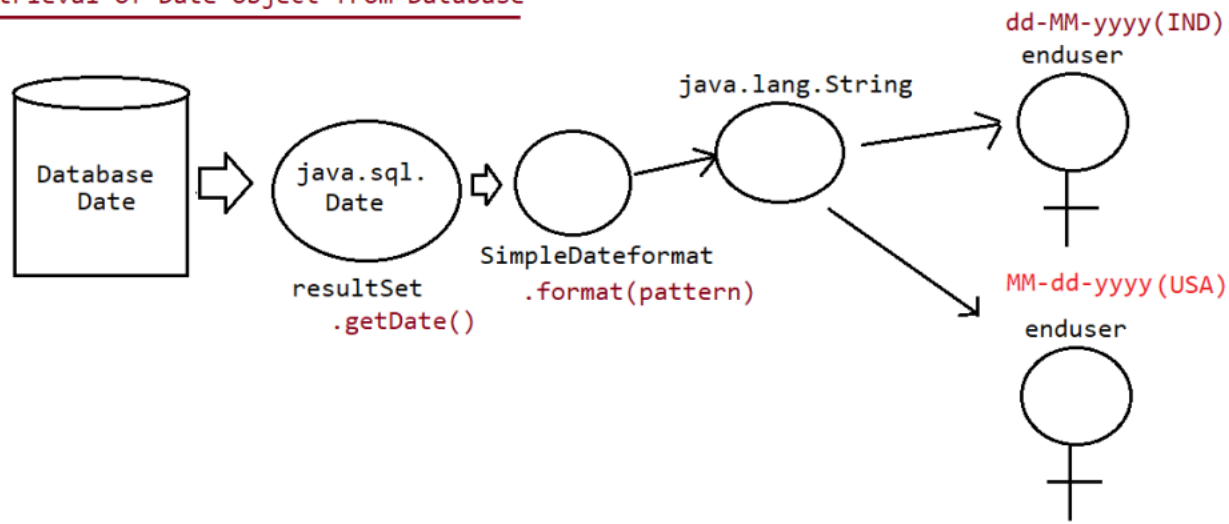
Need of DTO in projects
=======================
 DTO -> It stands for Data Transfer Object.
 This object is used for transferring the data from one layer to another layer in realtime applications.

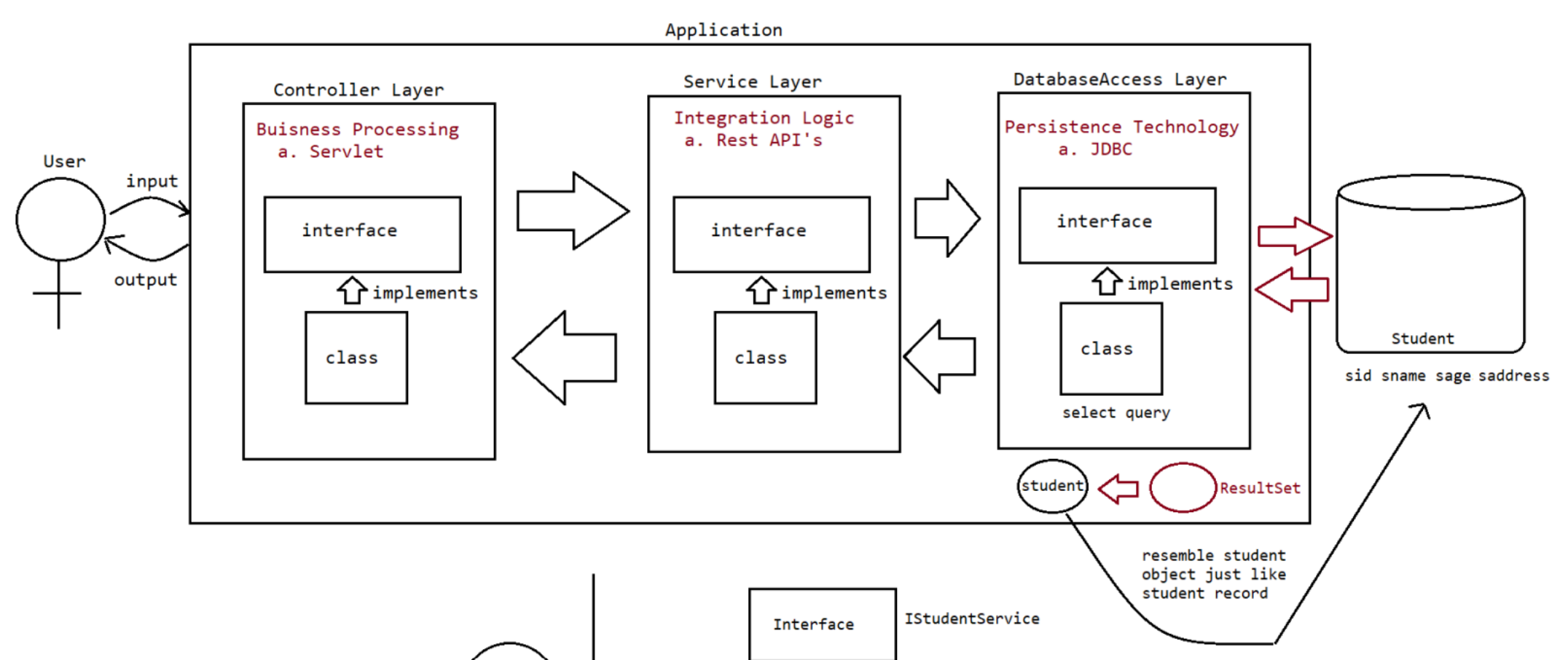

java.util.Date
(Date and Time)
extends
java.sql.Date
only Date
IND
user
dd-mm-yyyy
String
USA
user
yyyy-mm-dd
String
.java
Statement object
a. We should
code as
per DB format.
PreparedStatement
a.we need to
just supply
java.sql.Date
MySQL
Date
yyyy-mm-dd
Oracle
Date
dd-MMM-yy

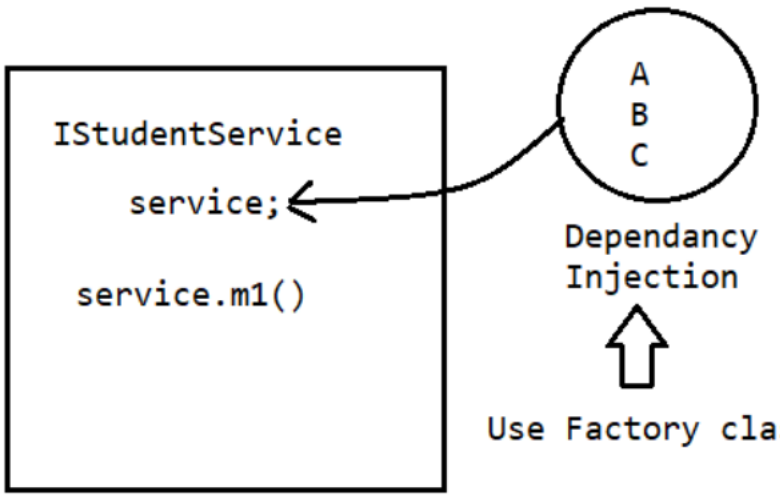

MM-dd-yy
dd-MM-yyyy
String format
MM-dd-yy
dd-MM-yyyy
SimpleDateFormat
.parse(string)
java.util.Date
. getTime()
java.util.Date
java.sql.Date
java.sql.Date
pstmt.setDate
(sqlDate)
pstmt.setDate
(sqlDate)
Oracle
dd-MMM-yy (03-JAN-93)
MySQL
yyyy-mm-dd (1993-03-01)

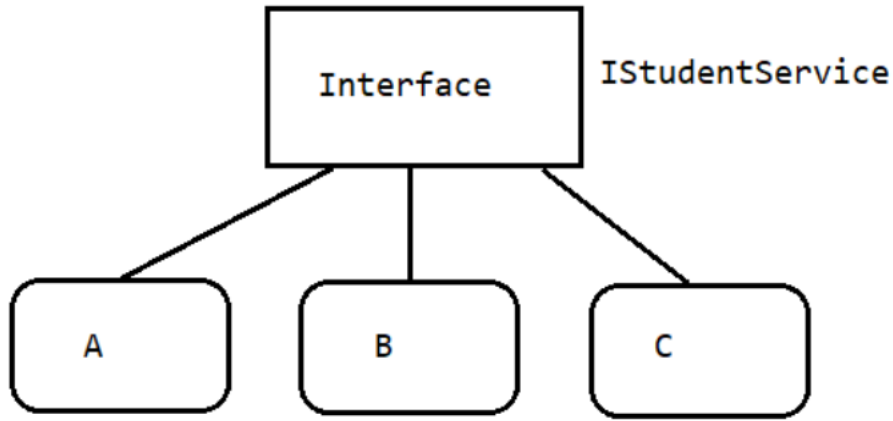


Retrieval of Date Object from Database




Application
Controller Layer
Buisness Processing
a. Servlet
interface
implements
class
Service Layer
Integration Logic
a. Rest API's
interface
implements
class
DatabaseAccess Layer
Persistence Technology
a. JDBC
interface
implements
class
select query
User
input
output
Student
sid sname sage saddress
student
ResultSet
resemble student
object just like
student record
Interface
IStudentService





Body of methods available in implementaion clas

A class, B class, C class.

m1()	m1()	m1()
m2()	m2()	m2()
m3()	m3()	m3()

Working with Large Objects (BLOB And CLOB)

Sometimes as the part of programming requirement,we have to insert and retrieve large files like images,video files,
audio files,resume etc wrt database.
Eg:upload image in matrinomial web sites
upload resume in job related web sites

To store and retrieve large information we should go for Large Objects(LOBs).

There are 2 types of Large Objects.
1. Binary Large Object (BLOB)
2. Character Large Object (CLOB)

1) Binary Large Object (BLOB)
A BLOB is a collection of binary data stored as a single entity in the database.
BLOB type objects can be images,video files,audio files etc..
BLOB datatype can store maximum of "4GB" binary data.

2) CLOB (Character Large Objects):
A CLOB is a collection of Character data stored as a single entity in the database.
CLOB can be used to store large text documents(may plain text or xml documents)
CLOB Type can store maximum of 4GB data.
Eg: resume.txt

Steps to insert BLOB type into database:
1. create a table in the database which can accept BLOB type data.
```
create table persons(name varchar2(10),image BLOB);
```
2. Represent image file in the form of Java File object.
```
File f = new File("sachin.jpg");
```
3. Create FileInputStream to read binary data represented by image file
```
FileInputStream fis = new FileInputStream(f)
```
4. Create PreparedStatement with insert query.
```
PreparedStatement pst = con.prepareStatement("insert into persons values(?,?)");
```
5. Set values to positional parameters.
```
pst.setString(1,"sachin");
```

To set values to BLOB datatype, we can use the following method: setBinaryStream()
```
public void setBinaryStream(int index,InputStream is)
public void setBinaryStream(int index,InputStream is,int length)
public void setBinaryStream(int index,InputStream is,long length)
```

6. execute sql query
```
pst.executeUpdate();
```

Steps to Retrieve BLOB type from Database

1. Prepare ResultSet object with BLOB type
```
ResultSet rs = st.executeQuery("select * from persons");
```

2. Read Normal data from ResultSet
```
String name=rs.getString(1);
```

3. Get InputStream to read binary data from ResultSet
```
InputStream is = rs.getBinaryStream(2);
```

4. Prepare target resource to hold BLOB data by using FileOutputStream

```
FileOutputStream fos = new FOS("katrina_new.jpg");
```

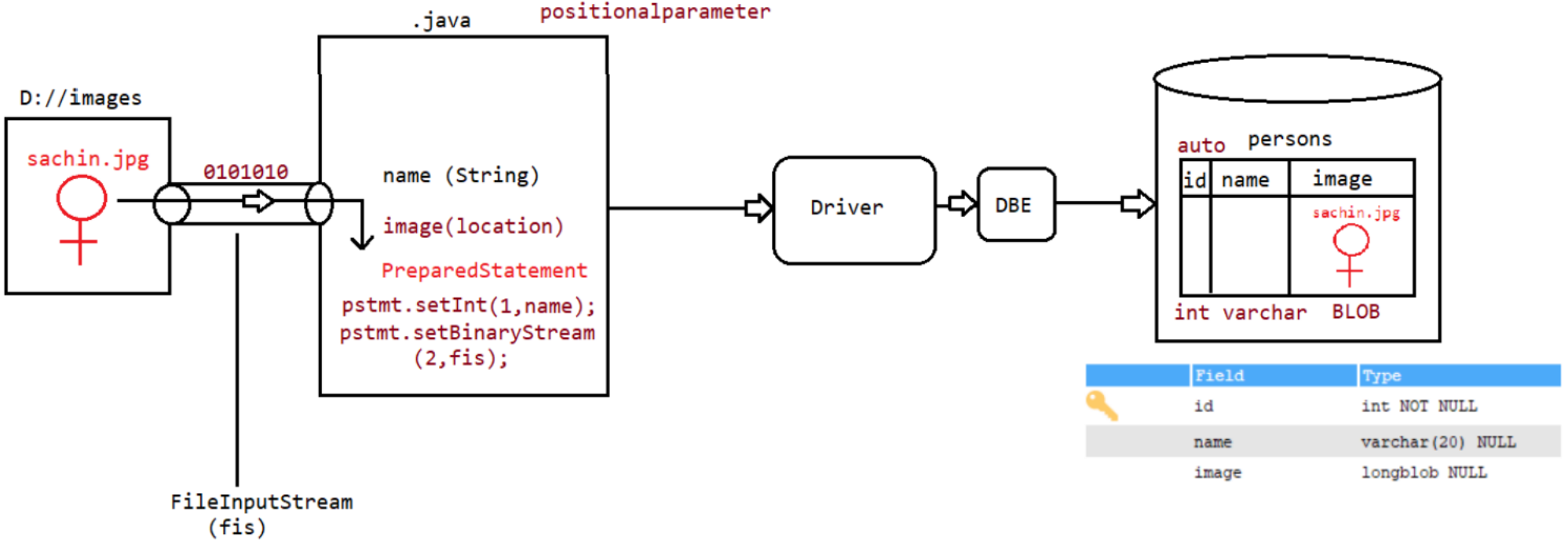
5. Read Binary Data from InputStream and write that Binary data to output Stream.

```
int i=is.read();
while(i!=-1)
{
      fos.write(i);
      is.read();
}

      or
byte[] b= new byte[2048];
while(is.read(b) > 0){
      fos.write(b);
  }
```

1,2

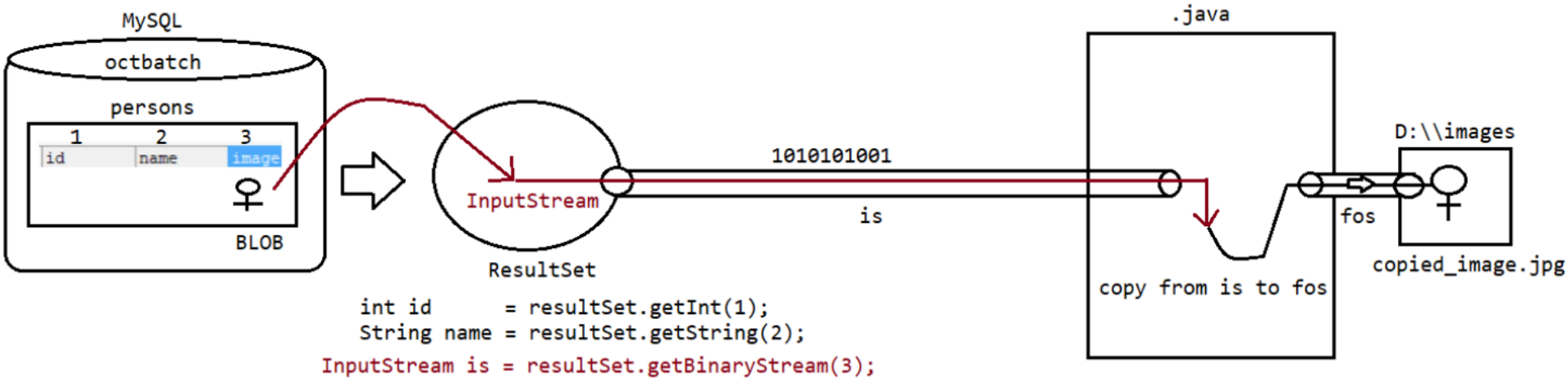
```
String sqlInsertQuery = "insert into persons(`name`,`image`) values (?,?)";
public abstract void setBinaryStream(int index, InputStream is) throws SQLException;
```

positionalparameter



	Field	Type
🔑	id	int NOT NULL
	name	varchar(20) NULL
	image	longblob NULL

Retreival Operation




```
int id      = resultSet.getInt(1);
String name = resultSet.getString(2);
InputStream is = resultSet.getBinaryStream(3);
```

Performance low

```
int i = is.read();
while (i != -1) {
   fos.write(i);
   i = is.read();
}
```

Performance is high,but difficult for developers

```
byte[] b = new byte[1024];
while (is.read(b) > 0)
{
     fos.write(b);
}
```

apache commons-io.jar

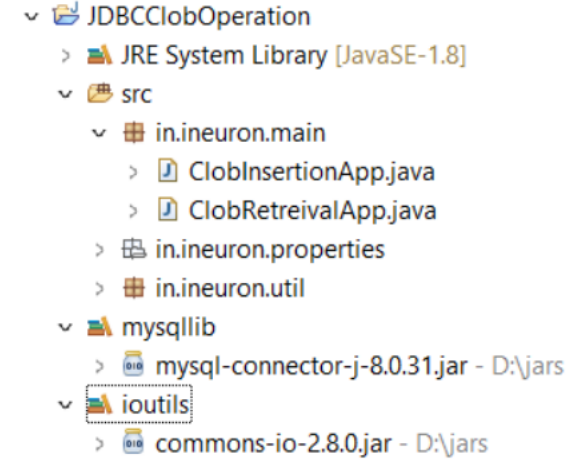
```
IOUtils.copy(is, fos);
```

clob operation

Field	Type
id	int NOT NULL
name	varchar(20) NULL
history	longtext NULL

```
void setCharacterStream(int parameterIndex,Reader reader) throws SQLException;
```

```
pstmt.setCharacterStream(2, new FileReader(new File(pdfLoc)));;
```



Retrieval Operation

```
Reader reader = resultSet.getCharacterStream(3);
File file = new File("history_copied.txt");
FileWriter writer = new FileWriter(file);
IOUtils.copy(reader,writer);
```

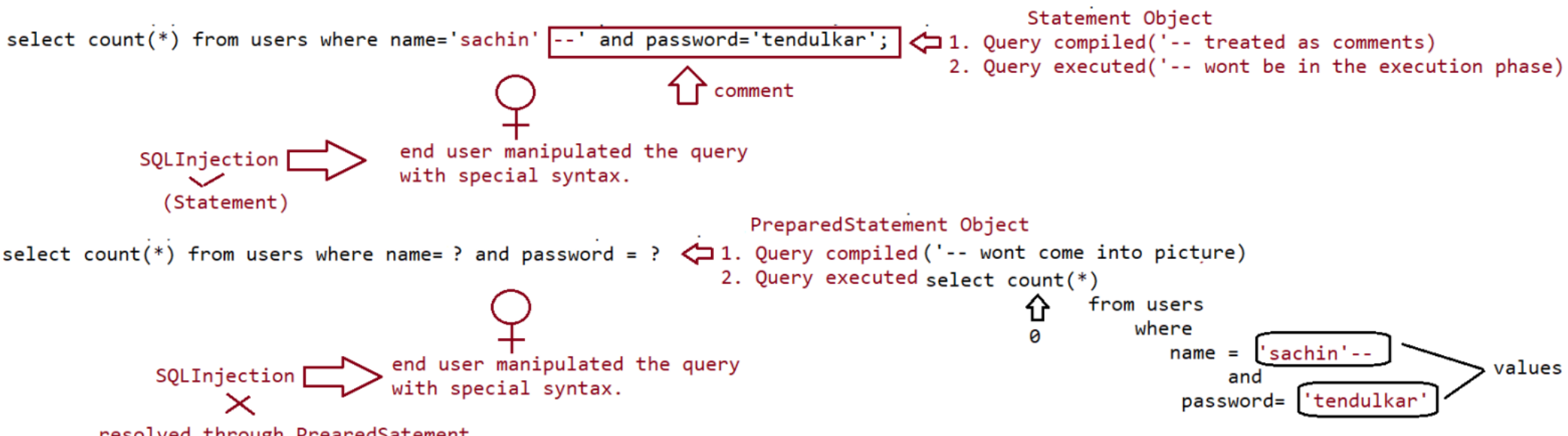
SQLInjection
============

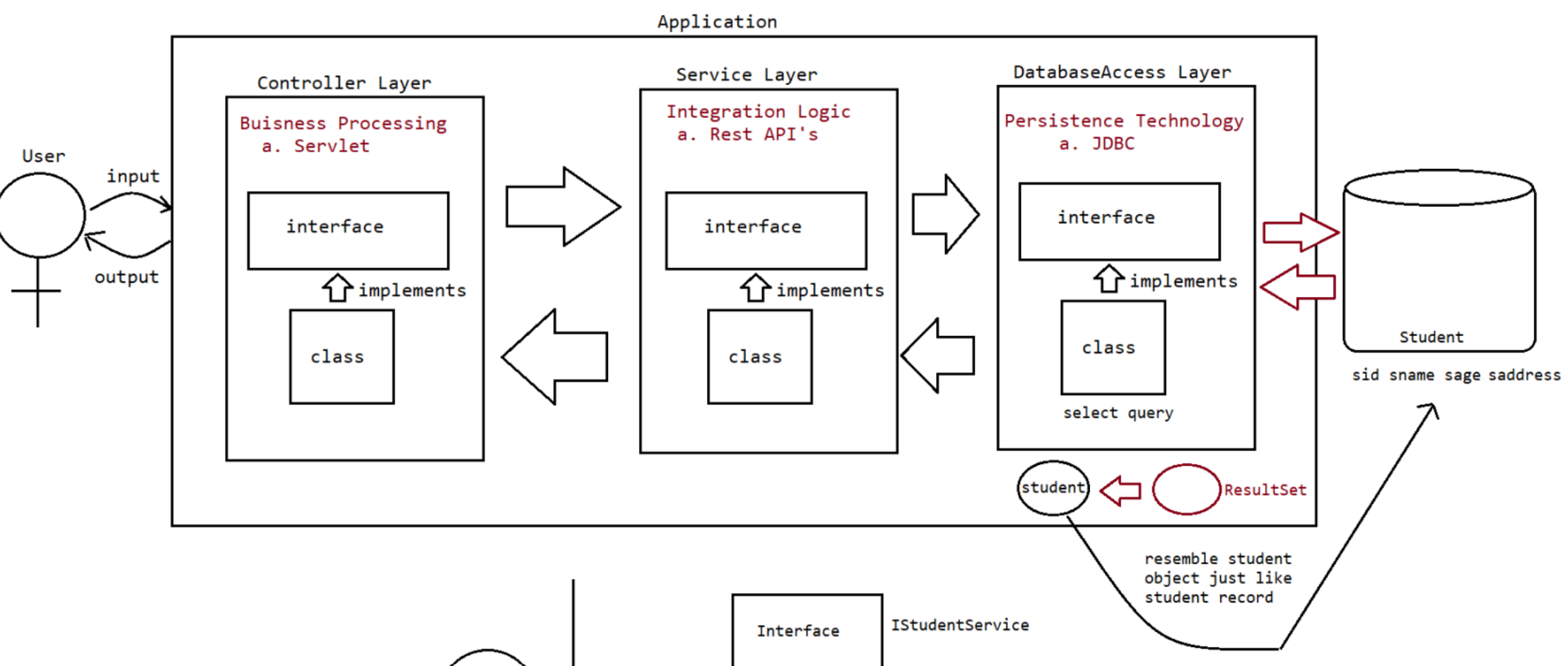
Comments in SQL are of 2 types

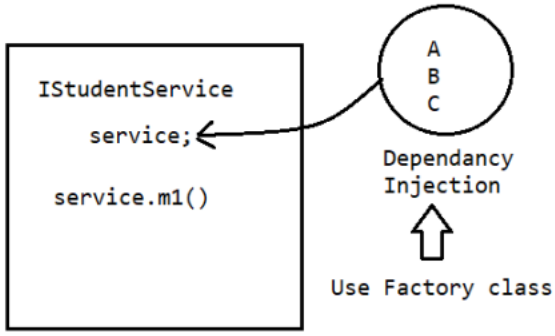
a. -- single line comment

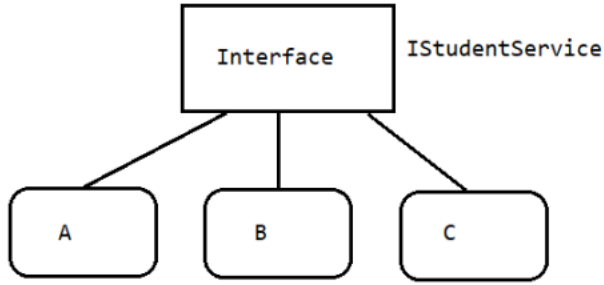
b. /*
multiline comment
*/

eg: select count(*) from users where name='sachin' and password='tendulkar'; => no problem with inputs




Application
Controller Layer
Buisness Processing
a. Servlet
interface
implements
class
Service Layer
Integration Logic
a. Rest API's
interface
implements
class
DatabaseAccess Layer
Persistence Technology
a. JDBC
interface
implements
class
select query
User
input
output
Student
sid sname sage saddress
student
ResultSet
resemble student
object just like
student record
Interface
IStudentService

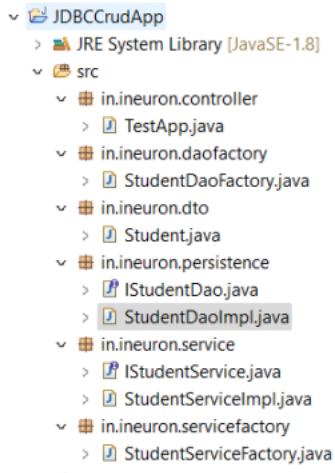




Body of methods available in implementaion clas

A class, B class, C class.

m1()	m1()	m1()
m2()	m2()	m2()
m3()	m3()	m3()



CallableStatement
==========
Statement(I) => Executing the query(Query should be written by java developer)
PreparedStatement(I) => Executing the query(Query should be written by java developer)
CallableStatement(I) => Executing the query(Query will be coded by DBA and present inside database)

If we want to execute only StoredProcedure can we use Statement/PreparedStatement?
Answer: No, we need to use "CallableStatement"

In programming if any code is repeatedly required,we can define the code inside the method and we can call that method
multiple times based on our requirement.
hence forth methods are best reusable component in programming.

Similarly in Database programming, if any group of sql statements is repeatedly required then we define those sql statements
in a single group and we call that group repeatedly based on our requirement.
This group of sql statements that peform a particular task is nothing but "Stored Procedure".
StoredProcedure is the best reusable component at database level.

StoredProcedure
It refers to group of sql statements that peform particular task.

These stored procedures are stored permanently in database for future usage and hence the name "Stored procedure".
Usually StoredProcedures are created by DatabaseAdmin(DBA)
Every database has its own language to create StoredProcedures
a. Oracle -> PL/SQL
b. MySQL -> Stored Procedure Language
c. MicrosoftSQLServer -> Transact SQL(TSQL)

Similar to methods stored procedures has its own parameters.
Stored Procedures has 3 parameters
a. IN parameters(to provide input values)
b. OUT parameters(to provide output values)
c. INOUT parameters(to provide and to collect ouput)

Storedprocedure example
==========
```
CREATE
    PROCEDURE `octbatch`.`P_GET_PRODUCT_DETAILS_BY_ID`(
                                    IN id INT,OUT NAME VARCHAR(20),OUT rate INT,OUT qnt INT)
      BEGIN
                  select pname,price,qty into name,rate,qnt from products where pid = id;
      END$$
DELIMITER ;
```

DBA will run the stored procedure as shown below
==========
To call the stored procedure we use the following syntax as shown below
```
CALL `P_GET_PRODUCT_DETAILS_BY_ID`(2,@name,@rate,@qty);
SELECT @name,@rate,@qty;
```

Syntax for calling stored procedure from java program
==========

```
      String storedProcedureCall="{CALL P_GET_PRODUCT_DETAILS_BY_ID(?,?,?,?)}";
      CallableStatement cstmt = connection.prepareCall(storedProcedureCall);
```

When jvm encounters the above line, jvm will send the call to database.
DB engine will check whether the sepcified stored procedure is available or not.
if it is available then it will will return CallableStatement object representing
that procedure.

Note:

eg: CALL P_GET_PRODUCT_DETAILS_BY_ID(?,?,?,?)
  setXXXX() -> availabe to take care of setting the input values as per the
DBengine datatypes.

```
                  eg: setInt(1,id)------------> int
                         setString(2,name)--> varchar
```

Before getting the value from the storedprocedure , we need to register the out
variables with java specific datatypes.

Registering the output variables of StoredProcedure
      a. To map the Java datatype and Database specific datatypes we need to use
some mechanism.
      b. The mechanism used is "JDBC Types" which is also known as "Bridge Types".

eg: java datatype -> int
      JDBC type   -> Types.INTEGER
      DB datatype   -> number

eg: java datatype -> String
       JDBC type   -> Types.VARCHAR
       DB datatype   -> varchar,varchar2

eg: java datatype -> java.util.Date
      JDBC type   -> Types.DATE
       DB datatype -> Date

 getXXXX() -> available to get the value from the DBE as per the DB specific
datatype

```
                        DB(varchar)------>JAVA(String)
```

Note
  To execute stored procedure we use execute().

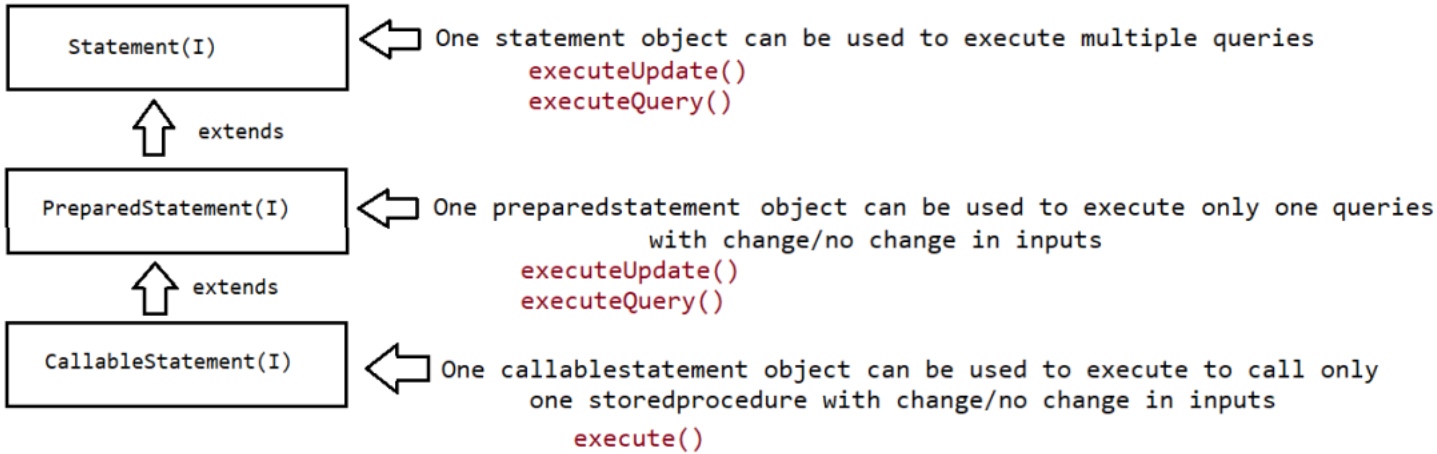
StoredProcedure to retrive all the records based on the product name
==========

```
DELIMITER $$
CREATE
    PROCEDURE `octbatch`.`P_GET_PRODUCT_BY_NAME`(IN name1 VARCHAR(20), IN name2
VARCHAR(20))
      BEGIN
            SELECT pid,pname,price,qty FROM products WHERE pname IN (name1,name2);
      END$$
DELIMITER ;
```

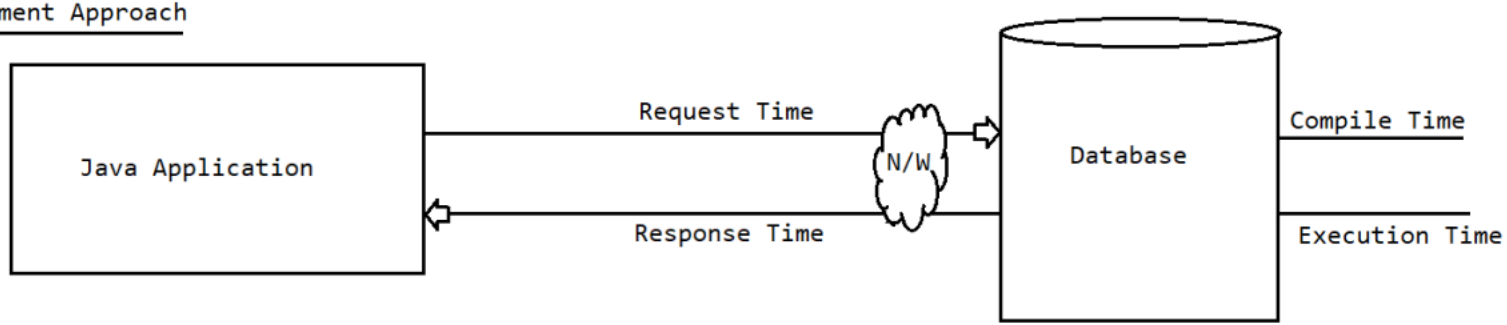
To call storedprocedure we use the following syntax

```
CALL `P_GET_PRODUCT_BY_NAME`('fossil','tissot');
```

```
output
  pid  pname   price     qty
   1     fossil      15000    2
   2     tissot     35000    3

                 refer: JDBCCallableStatementApp

BatchUpdate
===========
                refer: JDBCBatchUpdate
```

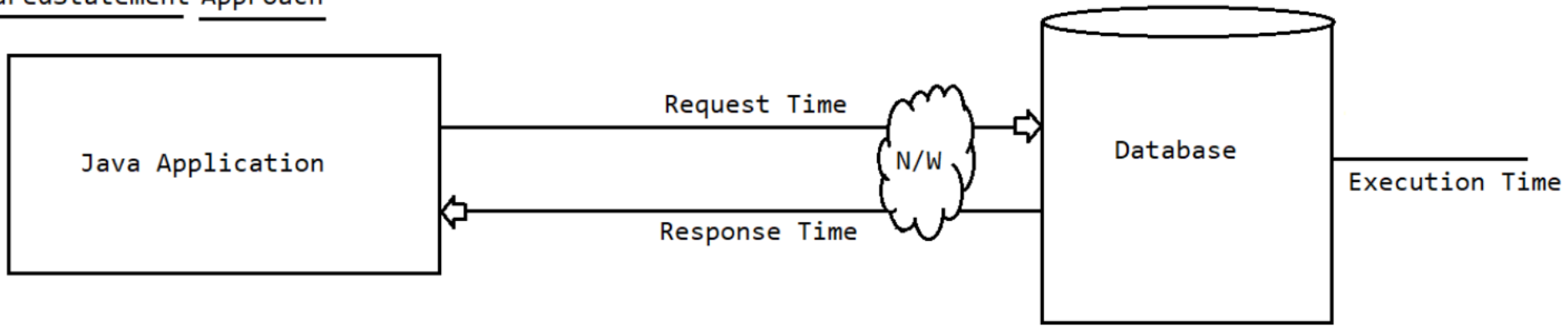


Statment Approach



```
Totaltime per query = R.T + C.T +  E.T + Resp.T
                    = 1ms + 1ms +  1ms + 1ms
                    = 4ms

For 1000 queries    = 1000 * 4
                    = 4000ms
```

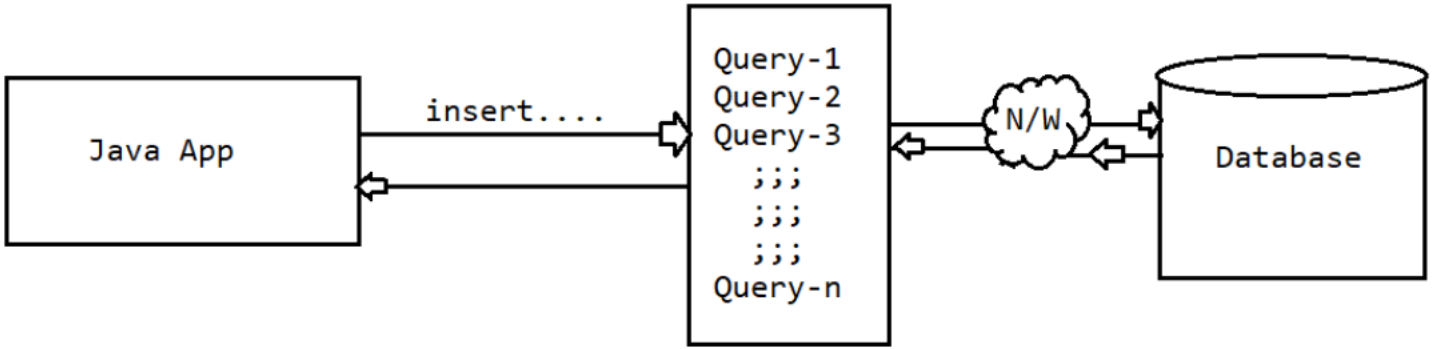
PreparedStatement Approach




```
Totaltime per query = R.T +  E.T + Resp.T
                    = 1ms +  1ms + 1ms
                    =  3ms

For 1000 queries    = 1000 * 3
                    = 3000ms
```

In the above 2 cases, we are trying to submit 1000 queries to the database one by one.
For 1000 queries to be submitted we need to hit the database 1000 times.
It increases the network traffic between java application and database.
It creates performance problems also.

To overcome the above problem, we should go for Batchupdates.
Grouping all the related queries into single batch and we can send that batch at a time to the database.



id	name	age	address
1	sachin	49	MI
2	dhoni	41	CSK
3	kohli	35	RCB
4	Gill	23	GT

Batch of queries

With Statement batch update

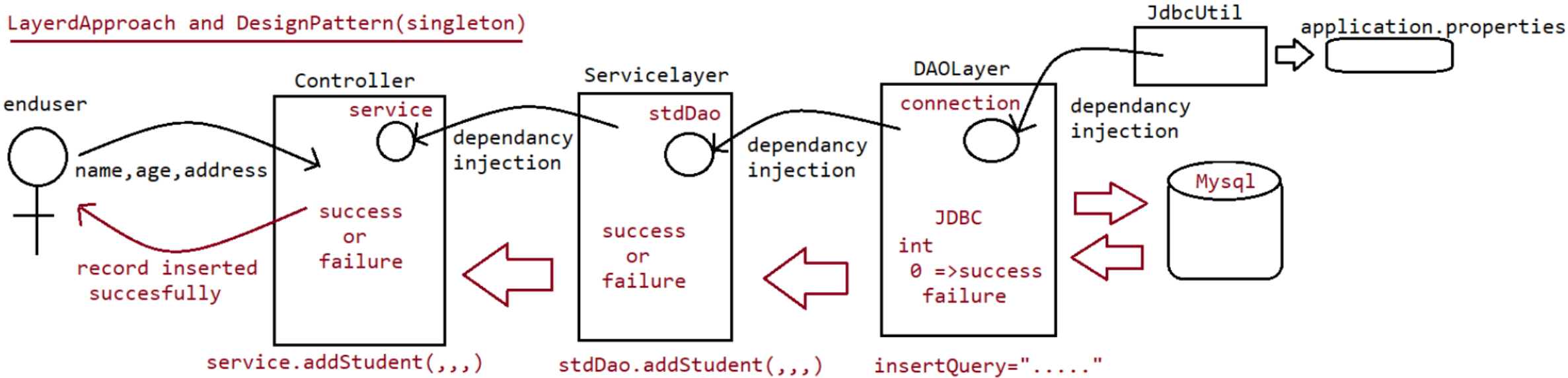
per 1000queries = R.T + 1000 C.T + 1000*E.T + Resp.T
= 1ms + 1000*1ms + 1000*1ms + 1ms
= 1ms + 1000ms + 1000ms + 1ms
= 2002 ms

With PreparedStatement batch update

per 1000queries = R.T + 1000*E.T + Resp.T
= 1ms + 1000*1ms + 1ms
= 1ms + 1000ms + 1ms
= 1002 ms

Advantage

a. reduce network traffic.
b. improve performance.


LayerdApproach and DesignPattern(singleton)
JdbcUtil
application.properties
DAOLayer
Servicelayer
Controller
enduser
connection
stdDao
service
dependancy injection
dependancy injection
dependancy injection
name,age,address
Mysql
JDBC
int
0 =>success
failure
success or failure
success or failure
record inserted succesfully
service.addStudent(,,,)
stdDao.addStudent(,,,)
insertQuery="....."

TransactionManagement

Process of combining all the related operations into a single unit and executing on the rule "either all or none" is referred as
"Transaction Management".

Case1 : Fund trasfer
1. debit funds from sender account (update query with bal=bal-amt)
2. credit funds into reciever account(update query with bal=bal+amt)

All operations should be peformed as single unit.
if money from sender account is debited, and if the money doesn't reach to receiver's account then there may be a
chance of   "Data inconsistency problem".

Case2: Movie Ticket Reservation
1. verify the status
2. Reserve the ticket
3. Payment
4. Issue ticket
All operations should be peformed as single unit.
If some operations success and some operations fails then there may be "Data inconsistency problem".

Transaction Properties

Every Transaction should follow the following four ACID properties

a. A -> Atomicity
        Either all operations should be done or none.
b. C -> Consistency
        It should ensure bringing the database from one state to another state.
c. I -> Isolation
        It ensures the transactions are isolated from other transactions.
d. D -> Durability
        It means once the transaction is commited,the results are permanent even in the case of system restart, errors,etc..

What are the types of Transactions available?
- a. Local Transaction
- b. Global Transaction

a. Local Transaction
        All operations in transaction are executed over the same database.
            eg: Funds transfer from one account to another account where the both the accounts in the same bank.

b. Global Transaction
        All operations in transaction are executed over the different database.
            eg: Funds transfer from one account to another account where the accounts are related to different banks.

Note:
- JDBC supports only local transactions.
- If we want global transactions then we need to go for frameworks like Spring/hibernate or EJB's.

Process of transaction management in JDBC

==========
1. Disable the autocommit nature of JDBC
      connection.setAutoCommit(false);

2. If all operations are completed means then we can commit the transaction using the following method
      connection.commit();

3. if any sql query fails, then we need to rollback the operations which are already completed using the following method
      connection.rollback();

SavePoint(I)
=========
 Within a transaction, if we want to rollback a particular group of operations based on some condition then we need to use
 savepoint
      a. getting savepoint using the following method
                  Savepoint sp = connection.setSavePoint();
        b. To perform rollback operation for a particular group of operations w.r.t savepoint then we need to use rollback .
                  connection.rollback(sp);
      c. we can release the savepoint or delete the savepoint as shown below
                   connection.releaseSavePoint(sp);

Case study
=========
connection.setAutocommit(false);
operation-1
operation-2
operation-3
SavePoint sp = connection.setSavePoint();
operation-4
operation-5

if(balance<1000)
      connection.rollback(sp);
else
      connection.releaseSavepoint(sp);

connection.commit();

if balance< 1000 then operation 4,5 will be rollbacked,otherwise all the operations will be commited.

eg:
      connection.setAutocommit(false);
            st.executeUpdate("insert into politicians values('BJP','Modi'));
            st.executeUpdate("insert into politicians values('TRS','KCR')");
            SavePoint sp = connection.setSavePoint();
            st.executeUpdate("insert into policitions values('BJP','siddu')");
            connection.rollback(sp);
                  ;;;;
                  ;;;
            connection.releaseSavePoint(sp);
                  ;;;
                  ;;;
            connection.commit();

Types of ResultSet
==========

refer: ****.png

absolute() -> it works from the BFR or from ALR.
relative() -> it works w.r.t current position.
In both the methods positive means move in forward direction, negative means move in backward direction.

Note:
   rs.last() and rs.absolute(-1) both are equal
   rs.first() and rs.absolute(1) both are equal

Note:

since resultSet holds differnt data work with different objectreference

Types of ResultSet

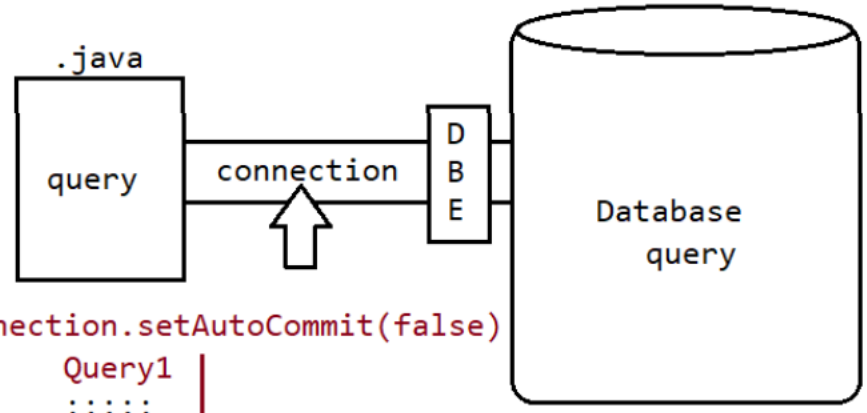
Divsion - 1(Based on the type of operation performed on ResultSet)
- a. Read only ResultSet(static ResultSet)
- b. Updatable ResultSet(dynamic ResultSet)

Divsion-2(Based on the movement of cursors)
- a. Forward only ResultSet(Non-Scrollable)
- b. Scrollable ResultSet
  - a. Scroll Insensitive ResultSet
  - b. Scroll Sensitive ResultSet

By default ResultSet object is
- a. static
- b. Non-scrollable

forward

	id	name	age	address
1	1	sachin	49	MI
2	2	dhoni	41	CSK
3	3	kohli	35	RCB
4	4	Gill	23	GT
5	5	sky	32	MI
6	6	jadeja	33	CSK
7	7	rohith	38	MI
8	8	rahul	31	LSG



since autocommit(false) is avaialble then, the query result will not be reflected in the database.

To reflect the result in the database we need to use
a. commit()
b. rollback()

By doing so we are following ACID properties.

#1
connection.setAutoCommit(false)

```
Query1
;;;;;
;;;;;
Query2
;;;;;
;;;;;
;;;;;
```

#2

```
if(......)
   connection.commit();
else
   connection.rollback();
```

#3 or #4 based on the condition

ResultSet types
============
Division-1
  a.Read only ResultSet(static resultset)
  b. updatable ResultSet(dynamic resultset)

Division-2
 a. NonScrollable ResultSet(forward only)
 b. Scrollable ResultSet
            1 In-Sensitive ResultSet
            2. Sensitive ResultSet

Scroll InSensitive ResultSet
      After getting resultSet if we are performing any operations in the database, and if those changes are not reflecting to
       the resultSet, such type of ResultSet are called as "Scroll Insensitive ResutlSet".
                  public static final int TYPE_SCROLL_INSENSITIVE

Scroll Sensitive ResultSet
      After getting resultSet if we are performing any operations in the database, and if those changes are getting reflecting to
       the resultSet, such type of ResultSet are called as "Scroll Sensitive ResutlSet".
                  public static final int TYPE_SCROLL_SENSITIVE

Note:
When we make the ResultSet as ScrollSensitive, then we need to use resultSet.refreshRow() to get the updated records
from the database.

Updatable ResultSet
=================
 It is possible to perform delete operation using resultSet without writing delete query
      resultSet.deleteRow() -> This method would delete the record based on the cursor position of ResultSet.

It is possible to perform insert operation using resultSet without writing insert query
      a. resultSet.moveToInsertRow()//creates a empty record.
      b. resultSet.updateXXXX(int pos,Object value);// insert the value based on the column data.
      c. resultSet.insertRow();// record will be inserted to the table with the updated values.

It is possible to peform update operation using resultSet without writing update query
      a. resultSet.getXXXX(int pos) // getting the old value from resultSet
      b. resultSet.updateXXXX(int pos, Object newValue); // update the newValue w.r.t resultSet
      c. resultSet.updateRow();// record will be updated to the table as per the resultSet information.

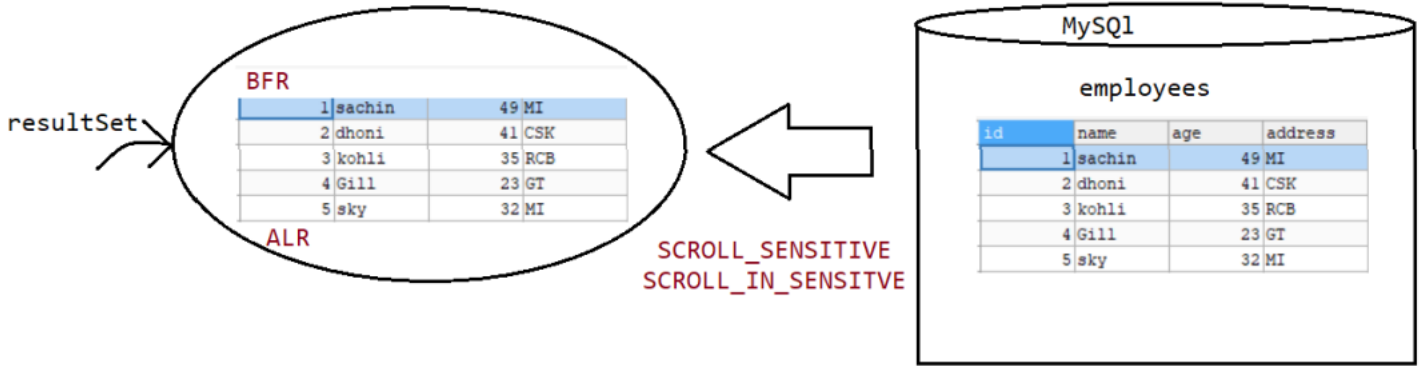
Working with Excel sheet

 To work with Excel sheet, we need to use HXTT company supplied driver we need to use.

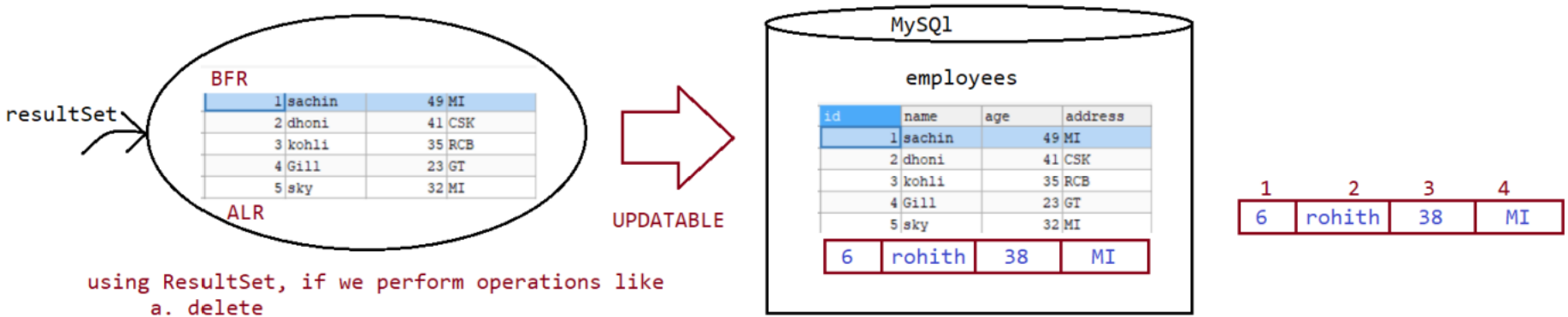
eg: EXCEL_JDBC40.jar

Working with CSV files

To work with CSV files, we need to use HXTT company supplied driver we need to use.
eg: Text_JDBC42.jar

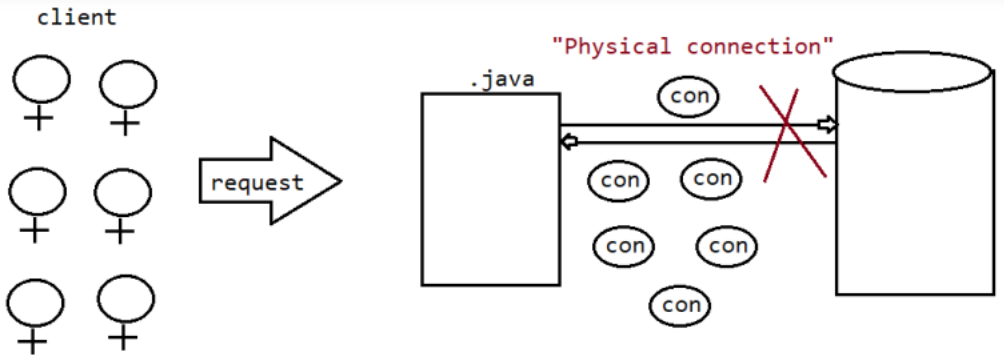


Assume some updations is happening on the records
in the database, will it be reflected automatically
in the same resultSet object or not?

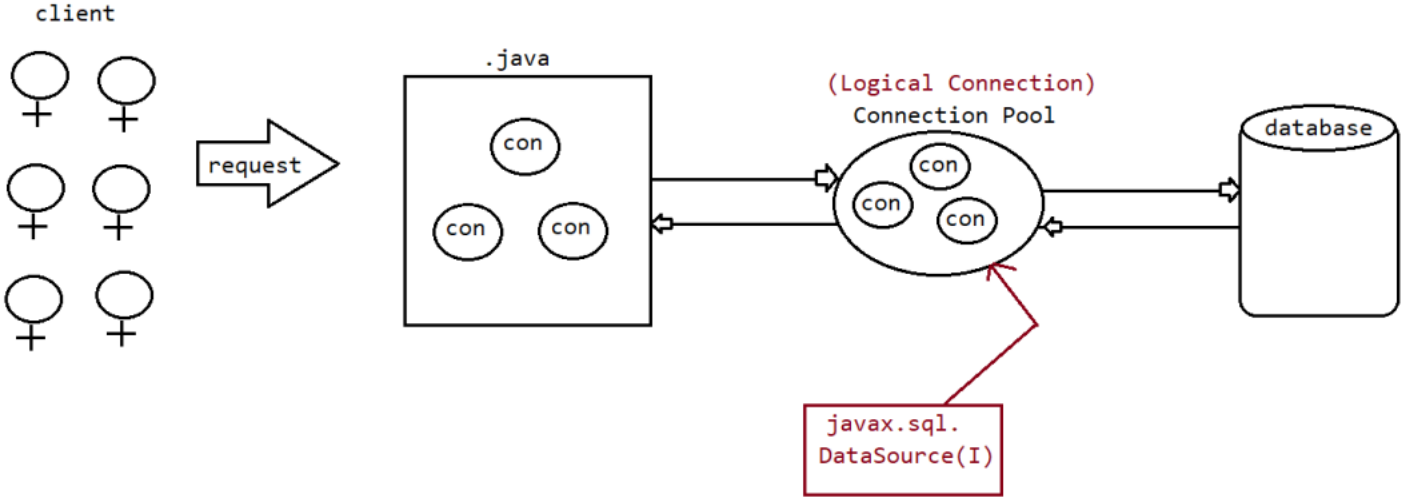


using ResultSet, if we perform operations like

a. delete
b. update
c. insert



performance is low
  a. creating connection object when user sends request,takes time.
  b. after using connection object,destory the connection object
     takes time.



Advantages
==========
1. We need not want to make clients wait to get the connection object, if connection object is available pick from connection pool and give the connection object for service.
2. After using the connection object, connection object will be returned back to connection pool so it supports reusability.

Note:

The no of objects created in the connection pool depends of the dbvendor and since the count of objects is less(10),it is not suitalbe for production environment.
DBVendors also supplied "Connection poooling" mechanism in the jars.
The best suited vendor who supplies connection pooling mechanism is "hikaricp". ⇦ "default supported in springboot"

For MySQL database to implement Connection pooling we need to use the class called "com.mysql.cj.jdbc.MysqlConnectionPoolDataSource"

monday -> javax.sql.RowSet and connection pool of hikaricp
project of JDBC(desing pattern)

tuesday -> ServletAPI