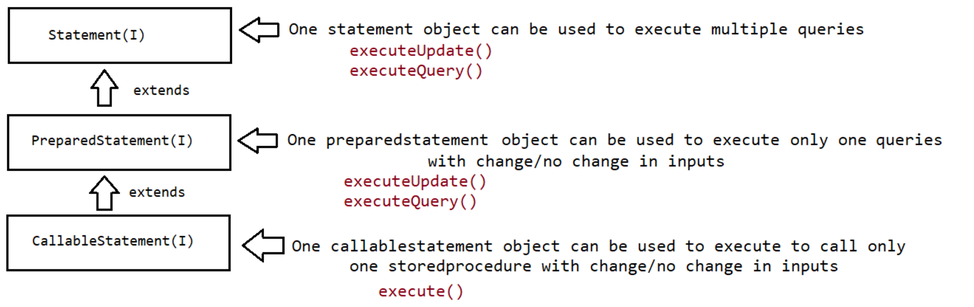
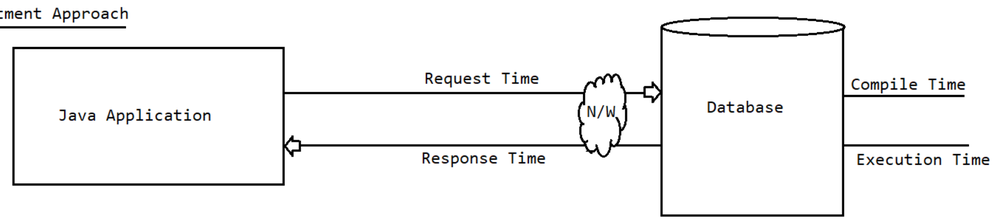
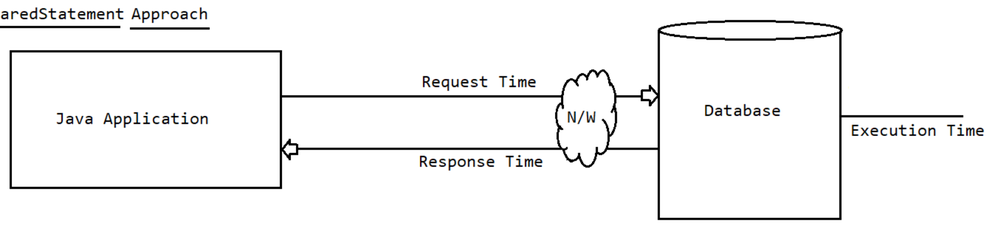


Statment Approach




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

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

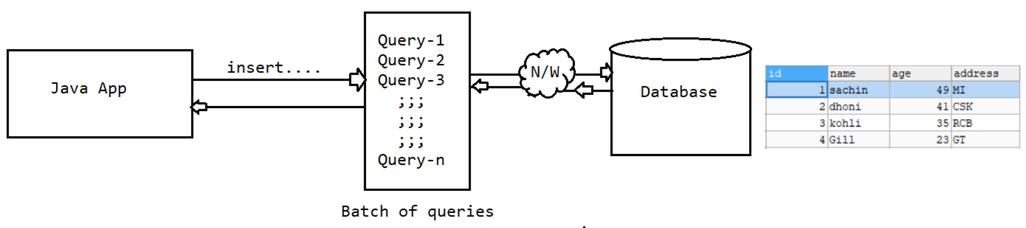
PreparedStatement Approach




```
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.

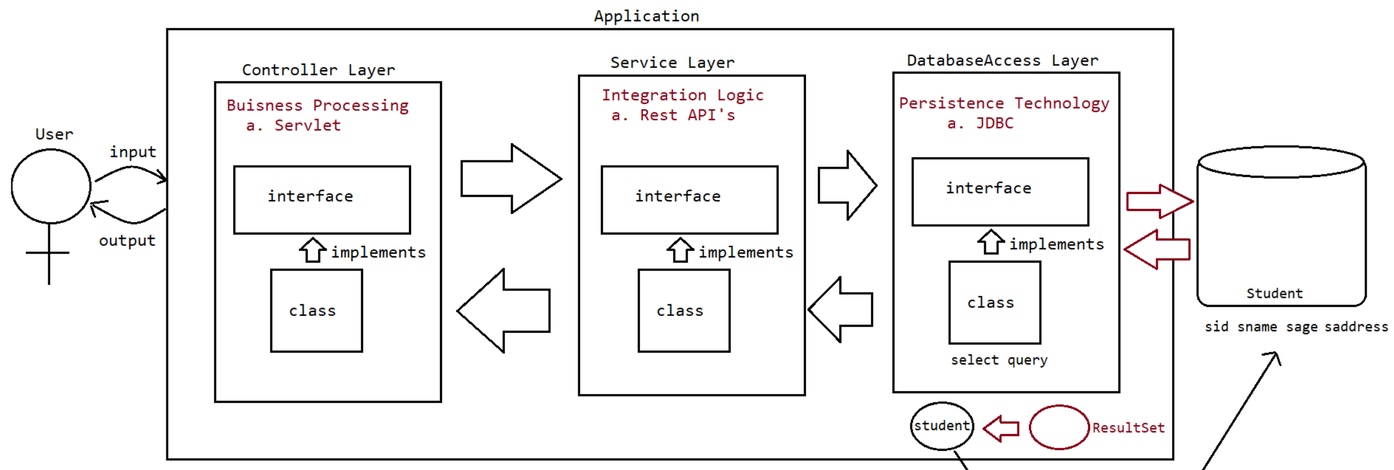


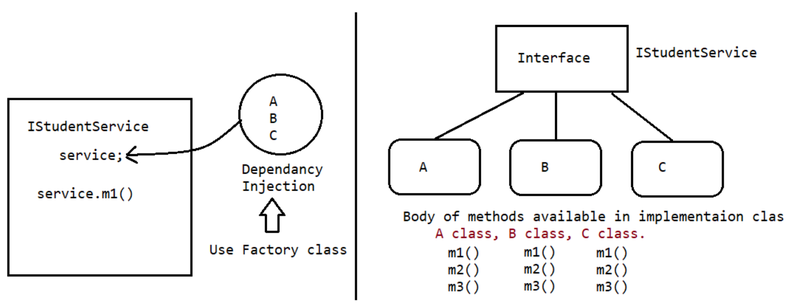
id	name	age	address
1	sachin	49	MI
2	dhoni	41	CSK
3	kohli	35	RCB
4	Gill	23	GT

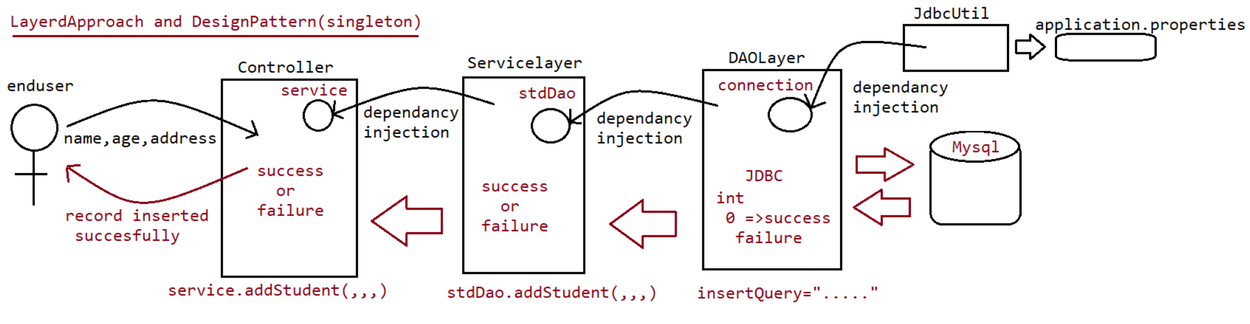
Batch of queries

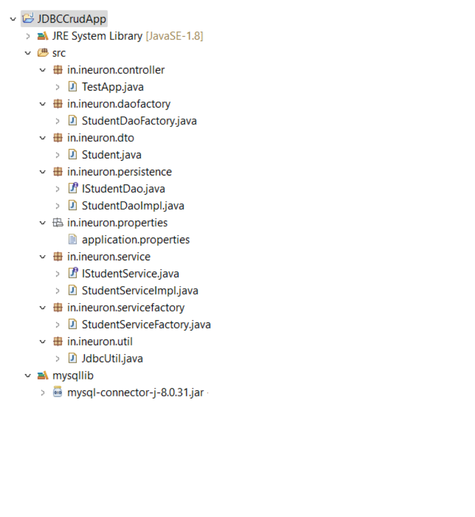
```
With Statement batch update
      per 1000queries =  R.T + 1000 C.T + 1000*E.T + Resp.T
                      =  1ms + 1000*1ms + 1000*1ms + 1ms
                      =  1ms + 1000ms   + 1000ms + 1ms
                      =  2002 ms


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.









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
```

```
Sample s =new Test();
   s.m1();
   s.m2();
   s.m3();
```

Exposing the set of services but hiding the method implementation is called "abstraction".