Layared Architecture in Java Based Business application:-
==========

1.maintaining the business data in secure and easily retrival manner.

--the logic that we write to implement this part of business application is known as Data Accees Logic.

2.processing the data according to the business rule .

--the logic that we write to implement this part of business application is known as Business/Service logic.

3.presenting the data to the user in user-understandable format.

--the logic that we write to implement this part of business application is known as Presentation logic.

--the above 3 logics is required for almost every business application.

```
class MyBusinessApplication{

//Data Access Logic  // fetch the account information from the DB

//Business Logic //calculating the Intreset to the balance

//Presentation Logic // display the information to the client


}
```

Note:- we can write all the 3 logics to develop a business application in one program/class itself , if we do so, the following problem we will face:-

1.all the logics to develop the application will be mixed up with each other (no clear code seperation).

2.modification done in one logic may affect the other logic .

3. logics will depend upon each other so, parellal developement will not be possible.

4.testing each logic is also will become complex..

--to solve this problem, a java based business application ,we devide into 3 logical partition .

and each part we say as a layer:-

1.Presentation Layer(UI layer)

2.Business Logic Layer (Service layer)

3.Data Access Layer

--a bussiness application will be devided into the logical partition depending upon the role played by each part.

--logical partition of a business application is known as layer.

Presentation Layer :-

--it is set of java classes, which are responsible for generating user input screen and response page(output screen) is knwon as PL.

--this layer provides the intraction with the end-user.

Business Logic Layer/Service Layer:-

--programatical implementation of business rule of a business organization is nothing but business logic .
--a collection of java classes whose methods have business logic to process the data according to the business rule is known as SL/BLL.

Data Access Layer :-

--a set of java classes whose methods r exclusivly meant for performing CRUD operation with the DB server is known as DAL.

using JDBC and DAO pattern

****Note:- to communicate among these layers loose coupling should be promoted.**

Developing Data Access Layer using ORM (Object Relational mapping) approach:-
==========

Java persistence:-

--the process of saving/storing java obj's state into the DB s/w is known as java persistence..

--for small application we can store business data (java object state) in the files using IO streams (serialization and deserailization approach).

--the logic that write to store java objs(which is holding business data) into the file using IO Streams is known as
"IO stream based persistence logic".

--but in the realtime application, we store/save/persist the business data inside the database using JDBC

public String saveStudentDetails(Student student)

--the logic that we write to store java objs data into the DB using JDBC is known as "Jdbc based persistence logic"

limitation of JDBC based persistence logic:-

1.jdbc can't store the java objs into the table directly,becoz sql queires does not allows the java objs as input, here we need to convert obj data into the simple(atmoic) value to store them in a DB.

2.jdbc code is the DB dependent code becoz it uses DB s/w dependent queries. so our jdbc based persistence logic is not 100% portable across various DB s/w.

3.jdbc code having boiler plate code problem (writing the same code except sql queries in multiple classes of our application again and again)..

4.jdbc code throws lots of checked exceptions, programmer need to handle them.

5.After the select operation, we get the ResultSet object.this RS obj we can not transfer from one layer to another layer and to get the data from the ResultSet we need to now the structure of the ResultSet.

6.there is no any caching and transaction management support is available in jdbc.

etc ...

--to overcome the above limitations we need to use ORM approach.

ORM (Object - ralation mapping):- Java----> relation
==========
--the process of mapping java classes with the DB tables ,, java class member variables with the DB table columns and
making the object of java class reperesents the DB table records having synchronization bt them is called a OR mapping.

student(roll, name, marks);

```
class Student{
roll;
name;
marks

}
```

Student s1=new Student(10,"Ram",500);

one Student object----------->one row of the student table

-here synchronization bt obj and table row is nothing but, the modification done in the obj will reflect the DB table row and vise-versa.

*****the logic that we write to store java objs into the DB using ORM approach is called as ORM based persistence logic.**

--there are variaous ORM s/w are available in the market, these s/w will act as f/w software to perform ORM based persistence logic.

ex:-

Hibernate ***
Toplink
Ibatis,
Eclipselink
etc...

f/w software :-

--it is a special type of s/w that provides abstraction layer on one or more existing core technology to simplyfy the process of application development.

--in java most of the f/w softwares comes in the form of jar files(one or more jar file)

--in order to use/work on these f/w softwares we need to add those jar files in our classpath.

--while working with the ORM based persistence logic we write all the logics in the form of objs without any sql quiery support. due to which our logic will become DB s/w independent logic.

--In ORM based logic, the ORM s/w takes objs as an input and gives objs as an output so no need to convert object data to the primitive values.

--ORM s/w addresses the mismatches bt object oriented representation of data and relational representation of data.

class <----> tables

class

1.inheritence mismatches / IS-A mismatch

2.Granularity mismatch / has-A mismatch

3. assotiation mismatch / table relation mismatch

i.e for processing and presenting the data, we represent the data in form of ObjectOriented fashion
whereas for storing the data we represnet the data in the form of relational fashion(in the tables)

class Student{

int roll,
String name,
int marks,

Address addr; // Has-A

}

class Address{
(city,state,pin)

}

student table :-

roll int
name varchar
marks int

address table :-

--one obj of Student class will represnt one row of student table

ORM s/w addresses these mismathces in very easy manner.

POJO class:- java bean class:

Plain old java object

--it is a normal java class not bounded with any technology or f/w s/ws.
i.e a java class that is not implementing or extending technology/framework api related classes or interfaces.

--a java class that can be compiled without adding any extra jar files in the classpath are known

as a POJO class.

POJI (plain old java interface)

Note:- every java bean class is a POJO but every POJO is not a java bean.

--the following class comes under the category of POJO class
class X implments Serializable, Runnable{

}

--this class does not comes under POJO
class X implements Servlet{

}

public class X {

public X(int x){

}

}

--above class is a POJO class but it is not a java bean class.becoz of parameterized constructor.

ORM s/w features:-
==========

1.it can persist/store java obj to the DB directly.

2.it supports POJO and POJI model

3.it is a lightweight s/w becoz to excute the ORM based application we need not install any kind of servers.

4.ORM persistance logic is DB independent. it is portable accross multiple DB s/ws. (beocz here we deal with object, not with the sql queires)

5.prevent the developers from boiler plate code coding to perform CRUD operarions.

6.it generates fine tuned sql statements internally that improves the performance.

7.it provides caching mechanism (maintaing one local copy to inhance the performance)

8.it provides implicit connection pooling.

9.exception handling is optional becoz it throws unchecked exceptions.

10.it has a special Query language called JPQL (JPA query language) that totally depends upon the '
objects. select eid,ename from employee.

sql> select roll, name, marks from student;

jpql> select roll, name,marks from Student;

--in sql we write the query in the term of tables and columns whereas in JPQL we write the Query in the term of classes and variables.

ORM

Hibernate "org.hibernate"
Toplink
Ibatis
EclipseLink "org.eclipse"

each ORM has thier own api to perform ORM based persistance logic.

(JPA)Java Persistence API: it is a standard api using which we can work with any kind of ORM s/w.

Hibernate has thier own api also,

Hibernate : ORM ---> diff classes , diff methods, diff interfaces

Ibatis : ORM ---> diff classes , diff methods, diff interfaces

sun-microsystem : JPA api (standard api)

--All the ORM s/w implements the JPA api.

Hibernate : JPA api

Ibatis: JPA api

--JPA api comes in the form of "javax.persistence" package.

Hibernate and JPA:-

JPA is a specification and Hibernate is its one of the famous implementation.

Hibernate:- it is one of the ORM based framework s/w. other s/w are :- toplink,ibatis,etc..

JPA:- (Java persistence api) :- it is an open specification given by Oracle corp, to develop any ORM based s/w .

JPA provides a standard api to work with any kind of ORM based s/w .

JPA api belongs from "javax.persistence" package.

--Hibernate is one of the most frequently used JPA implementation

--HB provides its own api to develop ORM based persistence logic , if we use those api then
our application will become vendor lock, ie we can not port our application accross multiple
ORM s/w.

--HB api comes in the form of "org.hibernate" package.

Note:- we get the JPA api , along with any ORM s/w , becoz all the ORM s/w implements

JPA specification.

java.sql
javax.sql this jdbc api comes along with jdk installation