Persistence logic
==========
The logic which is written to perform peristence operation is called "Persistence logic".
operations are CRUD/SCUD/CURD.

To write persistence logic injava we have technology and framework
a. technology => JDBC
b. framework => ORM tools like hiberante, jpa,springorm,springdatajpa(hotcake),.....

When we already have JDBC as persitence logic, what is the need to for ORM?
limitations of JDBC
a. If we use JDBC to develop persistence logic,we need to write sql queries by following the syntax of "Database".
DBQueries are specific to Database,this makes JDBC not portable across mulitple databases.
JAVA => WORA
JAVA + JDBC ==> WORA(not supported)

b. JDBC technology if we use and write a code, there would be a boiler plate code in our application.
Boiler plate code => A code which would repeat in multiple parts of the project with no change/small change is called boiler plate code.

CRUD
=====
1. Load and register the driver(automatic from JDBC4.X)
2. Establish the connection
3. Create PrepareStatement
4. Execute the Query
5. Process the ResultSet
6. Handle the Exception
7. Closing the Resource

Step1,2,3,6,7 boiler plat code becoz it is a common logic.

C. JDBC technology throws only one Exception called "SQLException',but it is a CheckedException which means u should have handling logic otherwise code would not compile.

```
a.    try{

      }catch(SQLException e){

      }
b. public static void main(String... args) throws SQLException{

    }
```

d. JDBC technology has only Exception class called "SQLException",so we don't have detailed hierarch of Exceptions related to different problems.

e. JDBC ResultSet object is not serializable, so we can't send it over the network, we need to use Bean/POJO to send the data over the network by writing our own logic.

f. While closing the jdbc connection object, we need to analyze the code allot otherwise it would result in "NullPointerException".

eg: Connection con = DriverManager.getConnection(url,user,password)
if(con!=null){.....}
closing the connection object should take place in "finally" block only.
To make the usage of AutoCloseable, we need to know the syntax of "try with Resource".

g. Java ====> OOP's based language
Assume we need to send Student object to database, can we write a logic of Database query at Object level if we use JDBC?
No, Not possible becoz DBqueries always expectes the value,but not the object directly.

h. JDBC doen't have good support of Transaction Management
a. local
b. global(no support in JDBC)

i. JDBC supports only positional parameters,it is difficult for the user to inject the values
It doesnot support namedparamaters.
String sqlInsertQuery = "insert into student(`name`,`email`,`city`,`country`) values(?,?,?,?)";
String sqlInsertQuery = "insert into student(`name`,`email`,`city`,`country`) values(:name,:email,:city,:country)";

j. To use JDBC, Strong knowledge of SQL is required.

h. JDBC does not supports versioning ,timestamp as inbuilt features
versioning:: keeping track of how many times record got modified.
timestamp:: keep track of when record was inserted and when lastly it was modified.

k.While developing persitence logic using JDBC, we can't enjoy oops features like
a. inheritance
b. polymorphism
c. composition
becoz jdbc does not allow objects as input values in sqlqueries.

Solution: To all the problems mentioned above we develpo persistence logic usin "ORM".
ORM tools are hiberante,eclipselink,ibatis,jpa,.....
ORM -> Object Relation Mapping