Framework :

- ↳ pre-defined template
- ↳ Makes Developer life Easy
- → minimize the writing of boiler plate code.

=> ORM = {object Relational mapping}

↓↓

Hibernate

limitations of JDBC :-

==>JDBC is database dependent, bcz query are data-base dependent, its indeirectly making java platform dependent which is against rule of WORA;

==> JDBC Exceptions are Checked Exceptions. Whenever we use JDBC in code we must either handle exception with try-catch or duck exception with throws;

==>If the database structure is modified after developing project using JDBC in it, our project will be affected bcz we have hard coded SQL queries.

==> Transaction support is not good, it supports local transaction not the global transaction.

==> JDBC supports positional parameters(?) not the named parameters (name)
setString(1, "Java");
we have to remember numbering of proper positional parameter

==> Strong knowledge of SQL , bcz we hard code SQL queries in JDBC(Queries are embedded in JAVA)

==>For every kind of DB related operation from Java to Db, we have to follow fixed number of common steps.

Create --> 7 steps
Read ==> 7 steps
Update==> 7 steps
Delete --> 7 steps

==> While you develop JDBC app, we cannot use features of OOPS,
example: INSERT INTO STUDENT(sid, sname, sage, smarks) values(?,?,?,?);
setInt(1, 1) , setInt(2, "Rohan") .....
Student st=new Student(1, "Rohan", 16, 44); INSERT INTO STUDENT(sid, sname, sage, smarks) values(st);

Bcz of above and many other limitations of JDBC we have a solution in form of ORM which Hibernate best suited for Java developers.

=> Object Mapping

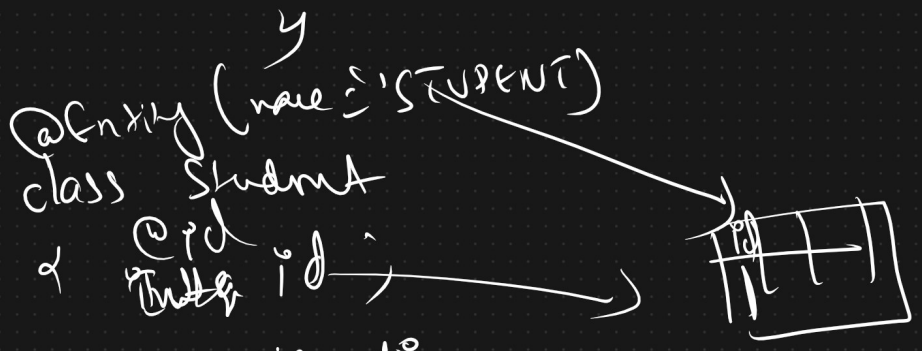
=> It's all about mapping/linking classes with db tables

=> Hibernate => (behind the scene -> JDBC)

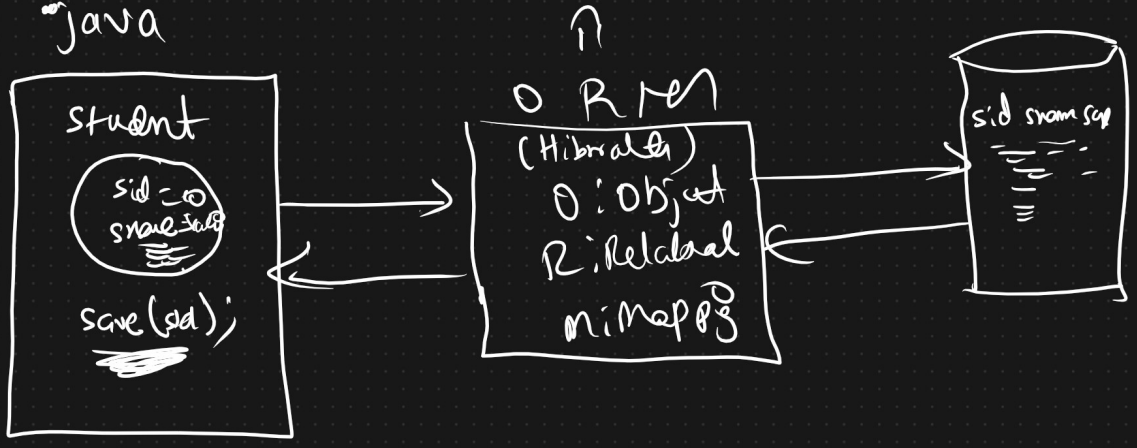
@Entity => class Student

@Id

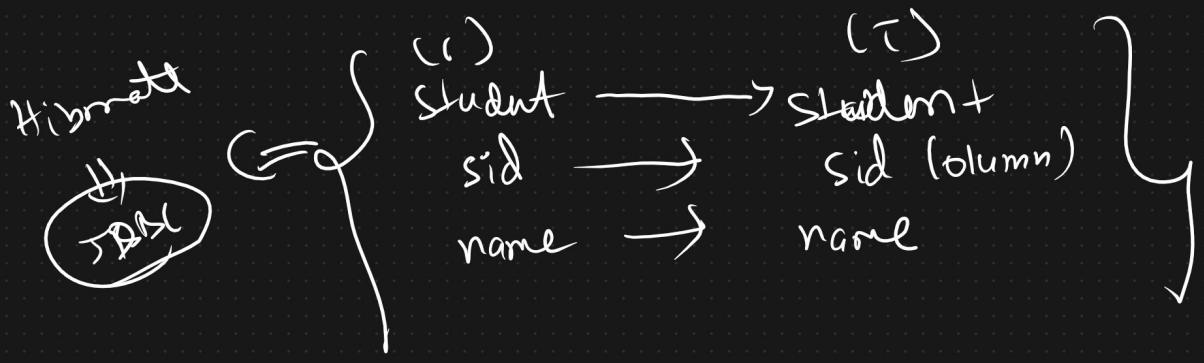
student



configuration
(xml approach
Annotation)



mapping mapping information


Hibernate
JPA
(1)
student
sid
name
(2)
student
sid (olumn)
name