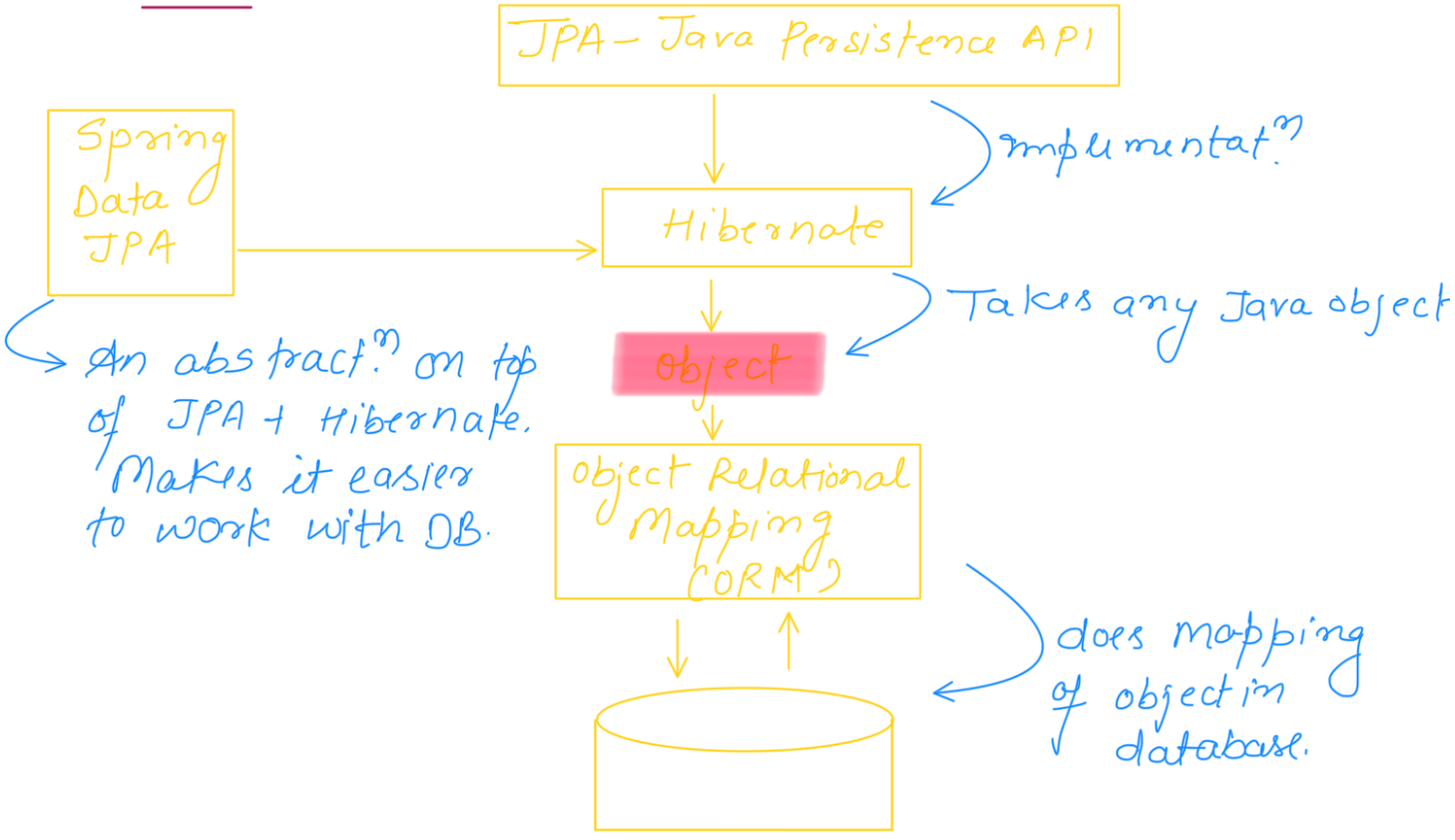
The Real Backend

JPA + HIBERNATE + Entity-Repo - Service - Controller.

- JPA → Java persistence API
- JPA implementation → inside springboot
- Entity → @Entity, @id, @generatedvalue.
- Repo → interfaces that talks to repo.

→ for today we will use H2 in-memory DB. which comes bundled inside SpringBoot.
→ Later we will be using mysql in near future.

JPA



Hibernate

→ converts code into queries for DB.

Eg → Code:
```
@Entity
class Student{
    Long id;
```

String name;
int age; }

hibernate converts this to

Student

| id | name | age |

→ We didn't write sql, Hibernate handled that.

① @Entity → Put this on a class to convert that into table.

② @ id → This is the primary key of a table.
Eg – @ Id
private long Id;

③ @ GeneratedValue – "Generate Id automatically"
→ for using this you need @ Id on top of it.

Eg – @ Id
@ GeneratedValue (Strategy = GenerationType = Auto)
privat long Id;

code
{ name : "Pankaj",
age : 21 } →

Hibernate
Insert into student (id, name, age)
values (1, 'Pankaj', 21)
↑ created by hybernate

When you run Spring Boot with JPA:

1. Hibernate scans for classes with **@Entity**
2. Reads the fields (id, name, age...)
3. Checks/creates the table
4. Matches Java fields → SQL columns
5. Manages relationships (OneToMany, ManyToOne, etc.)
6. Converts Java objects ↔ Database rows

This is called **ORM (Object Relational Mapping)**.

You write Java.

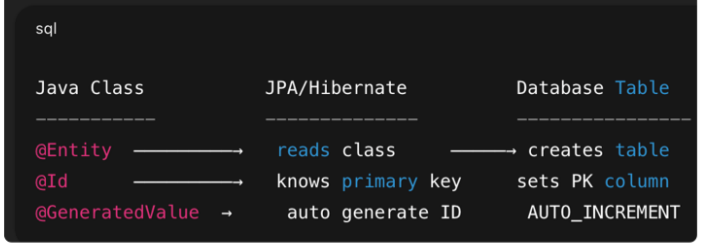
Summary

@ Entity – Make this class a DB table

@ Id – Set as primary key

Hibernate writes SQL.
Life becomes easy.

🎯 Understand in One Diagram



@GeneratedValue - Auto Increment above Id.