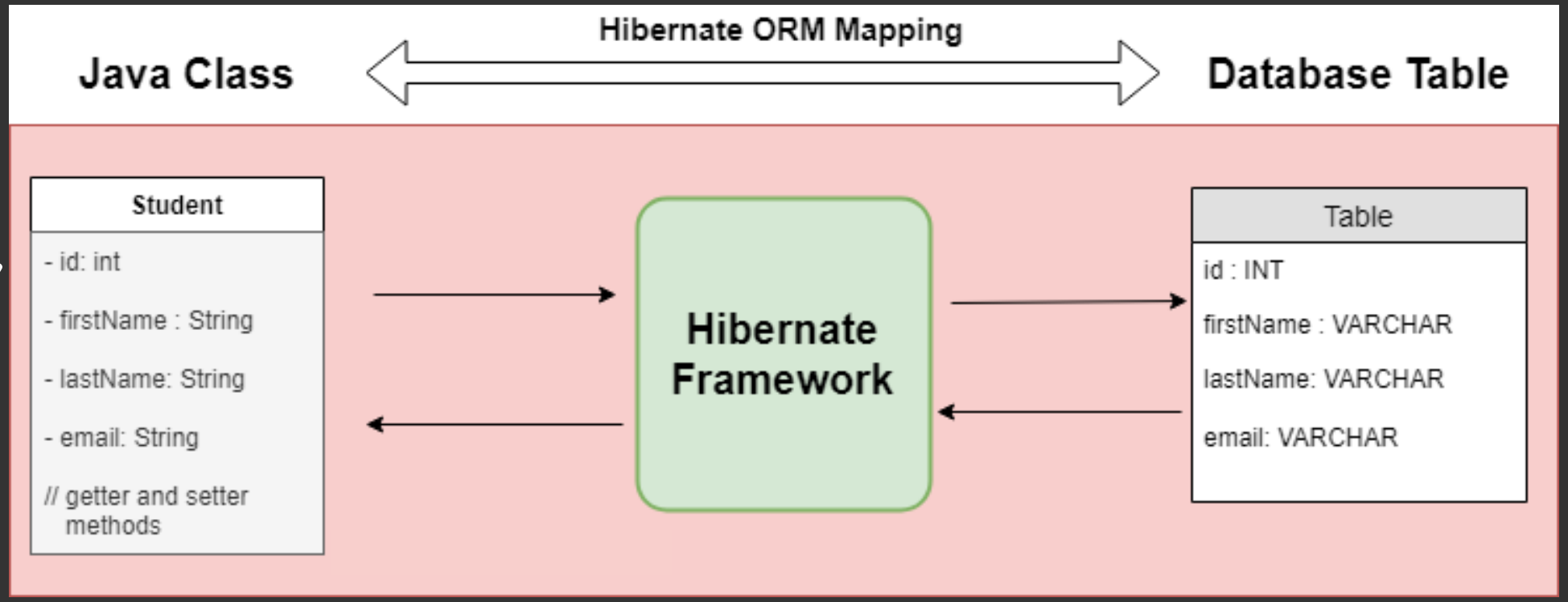
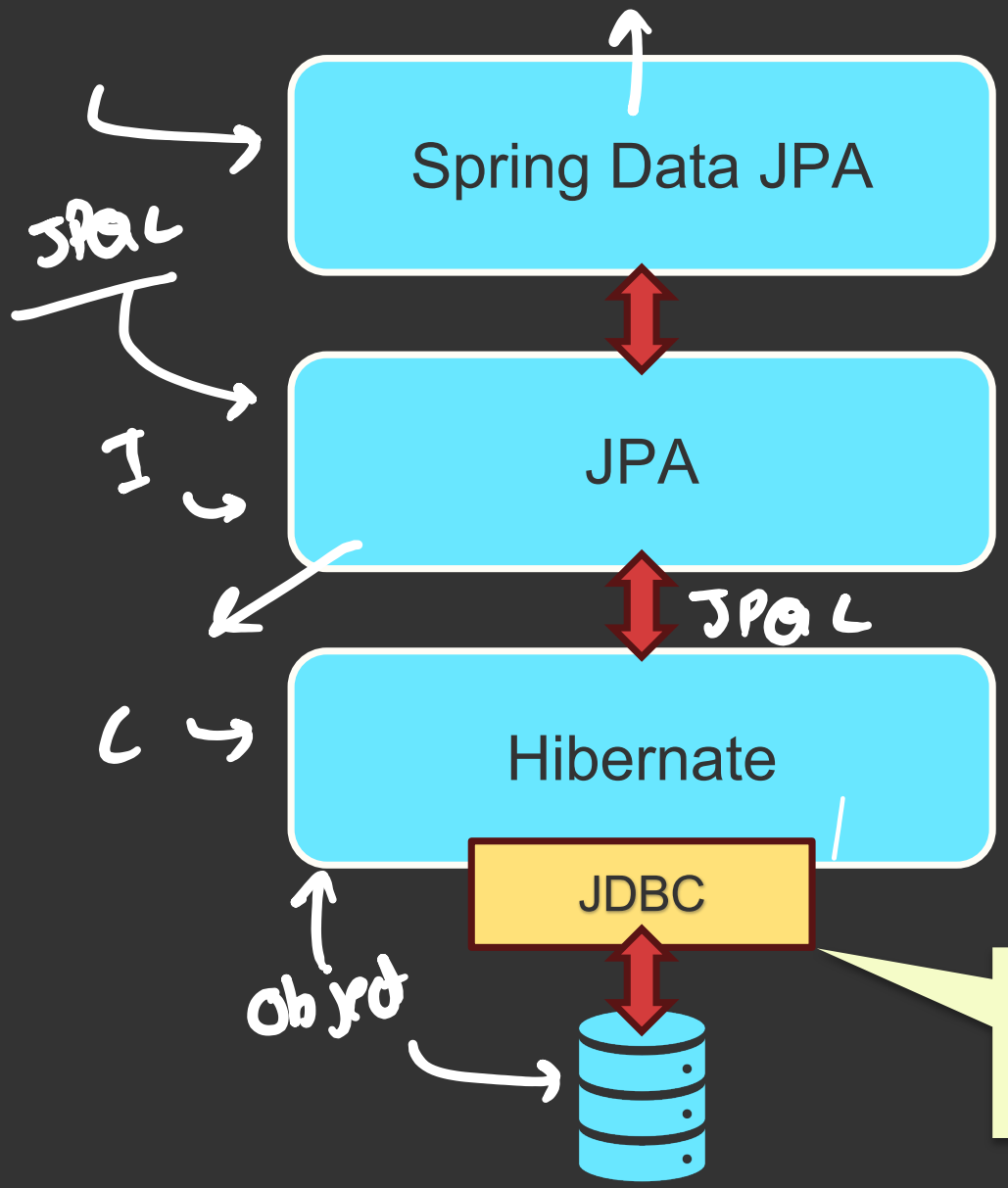
3.2

Hibernate and JPA

Hibernate ORM Mapping





Built on top of JPA and helps building DAO methods using Repository

Just a specification as to how the Relation and Objects mapping shout take place

Hibernate is the implementation built on JPA specification

Queries generated by hibernate get executed using JDBC

Hibernate

EM ←

Hibernate is a powerful, high-performance Object-Relational Mapping (ORM) framework that is widely used with Java. It provides a framework for mapping an object-oriented domain model to a relational database.

Hibernate is one of the implementations of the Java Persistence API (JPA), which is a standard specification for ORM in Java.

JPA (Java Persistence API)

JPA is a specification for object-relational mapping (ORM) in Java.

It defines a set of interfaces and annotations for mapping Java objects to database tables and vice versa.

JPA itself is just a set of guidelines and does not provide any implementation. The implementation of JPA is provided by ORM frameworks such as Hibernate, EclipseLink, and OpenJPA.

So Basically...

JPA Provides a standard for ORM in Java applications, ensuring that developers can switch between different JPA providers without changing their code.

And

Hibernate is one such JPA Provider.

However,

Hibernate is a specific implementation of JPA and a powerful ORM framework on its own. It offers additional features and optimizations beyond the JPA specification, making it a popular choice for ORM in Java applications.

Common Hibernate Configurations

1. spring.jpa.hibernate.ddl-auto=update/create/validate/create-drop/none
2. spring.jpa.show-sql=true
3. spring.jpa.properties.hibernate.format_sql=true
4. spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect (Optional)

Entity Annotation

- `@Entity`
- `@Table`
- `@Id`
- `@GeneratedValue(strategy = GenerationType.IDENTITY)`
- `@Column(name = "name", nullable = false, length = 50)`
- `@CreationTimestamp and @UpdateTimestamp`

Table Annotation

```
@Table(
    name = "employees",
    catalog = "employee_catalog",
    schema = "hr",
    uniqueConstraints = {
        @UniqueConstraint(columnNames = {"email"})
    },
    indexes = {
        @Index(name = "idx_name", columnList = "name"),
        @Index(name = "idx_department", columnList = "department")
    }
)
```

Key features of JPA

1. Entity Management: Defines how entities (Java objects) are persisted to the database.

2. Query Language: Provides JPQL (Java Persistence Query Language) for querying entities.

3. Transactions: Manages transactions, making it easier to handle database operations within a transactional context.

4. Entity Relationships: Supports defining relationships between entities (e.g., One-to-One, One-to-Many, Many-to-One, Many-to-Many).