ORM

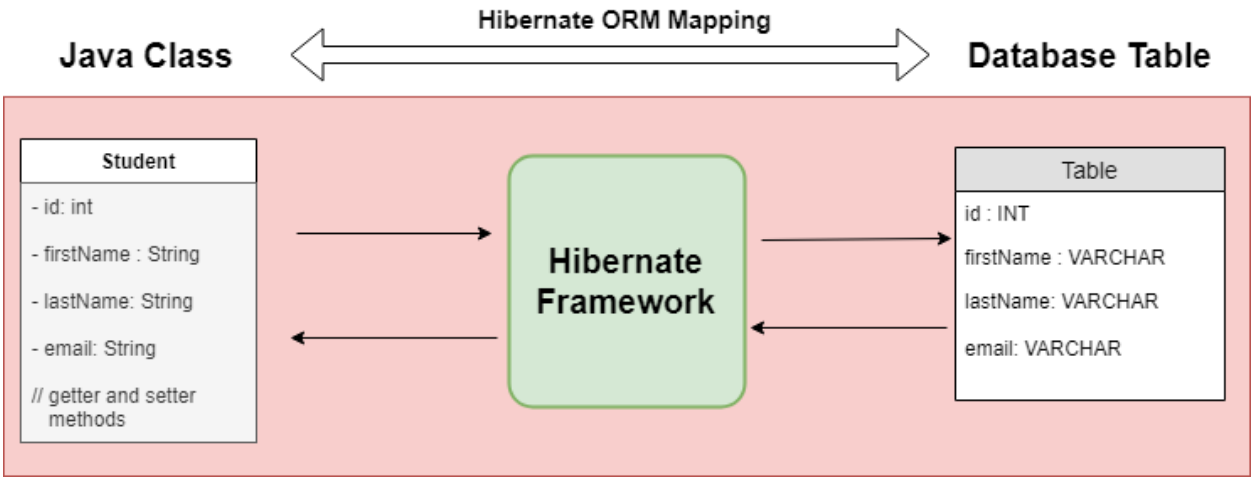
Object-Relational Mapping (ORM) is a technique that lets you query and manipulate data from a database using an object-oriented paradigm.

From a developer's perspective, an ORM allows you to work with database-backed data using the same object-oriented structures and mechanisms you'd use for any type of internal data.

In general, ORMs serve as an abstraction layer between the application and the database. They attempt to increase developer productivity by removing the need for boilerplate code and avoiding the use of awkward techniques that might break the idioms and ergonomics that you expect from your language of choice.

ORM hides and encapsulates change in the data source itself, so that when data sources or their APIs change, only ORM needs to change to keep up—not the applications that use ORM to insulate themselves from this kind of effort.

This capacity lets developers take advantage of new classes as they become available and also makes it easy to extend ORM-based applications. In many cases, ORM changes can incorporate new technology and capability without requiring changes to the code for related applications.



Following are the various frameworks that function on ORM mechanism: -

- Hibernate
- TopLink
- ORMLite
- iBATIS
- JPOX

JPA VS Hibernate:

JPA	Hibernate
JPA is described in **javax.persistence** package.	Hibernate is described in **org.hibernate** package.
It describes the handling of relational data in Java applications.	Hibernate is an Object-Relational Mapping (ORM) tool that is used to save the Java objects in the relational database system.
It is not an implementation. It is only a Java specification.	Hibernate is an implementation of JPA. Hence, the common standard which **is** given by JPA is followed by Hibernate.
It is a standard API that permits to perform database operations.	It is used in mapping Java data types with SQL data types and database tables.
As an object-oriented query language, it uses **Java Persistence Query Language (JPQL)** to execute database operations.	As an object-oriented query language, it uses **Hibernate Query Language (HQL)** to execute database operations.
To interconnect with the entity manager factory for the persistence unit, it	To create Session instances, it uses **SessionFactory** interface.

JPA	Hibernate
uses **EntityManagerFactory** interface. Thus, it gives an entity manager.	
To make, read, and remove actions for instances of mapped entity classes, it uses **EntityManager** interface. This interface interconnect**s** with the persistence condition.	To make, read, and remove actions for instances of mapped entity classes, it uses **Session** interface. It acts as a runtime interface between a Java application and Hibernate.