**ORM:** Framework which helps to convert java objects into database objects.

**JPA:** Specification which tells how to store the java objects into database

**Hibernate:** Implementation of JPA

Annotations ­will be in the form of interfaces. Attributes are like methods of it. Values taken by attributes are like return type of the methods.

**@Entity:** Represents pojo/java bean as a SQL table

**@Id:** Primary key of an entity

**@GeneratedValue:** Represents the primary key generation strategy. It will be ***AUTO*** by default

Attributes are ***strategy*** and ***generator***

1. ***strategy*** *->* **AUTO, *IDENTITY, TABLE, SEQUENCE***

* If we specify **AUTO**, the JPA provider will use any strategy it wants to generate the identifiers
* ***IDENTITY->*** Auto generates the primary key using identity column

1. ***generator->***Representssequence name

**@Table:** Representstable name, if we don’t specify it, table name will be same as entity name by default. we can mention the schema name using ***schema***attribute

**@Column:** Represents column name, if we don’t specify it, column name will be same as field name by default

Attributes are *insertable, updatable, nullable, unique, length, precision, scale*

1. ***insertable, updatable, nullable, unique –>***To represent respected SQL operations can be applied on column
2. ***length ->***Applies only if the string valued column is used, default value is 255, default value is 0.
3. ***precision, scale ->***Applies only if the decimal valued column is used, default value is 0.

**@Transient:** Represents non-persistent variable

**@Temporal:** Can be used only on java.util.Date or java.util.Calendar types. To format the data

**Hibernate Entity Life Cycle:** Transient, Managed, Detached and Deleted

**Persistence Context:** An implementation of **UnitOfWork**, which keeps track of changes made to the loaded data, eventually synchronize the changes back to the database at the end of business transaction. It’s implementations are Hibernate **Session** and **EntityManager**

1. **Managed Entity:** Data may or may not be present in database and managed by current running session and all the changes are propagated to database automatically

Refer (https://www.baeldung.com/hibernate-entity-lifecycle)

1. **Transient Entity:** Data will not be saved in database until we call save();converted to managed entity/persisted entity. It is not managed by any persistent context. Example is instantiating an object via it’s constructor
2. **Detached Entity:** It is not managed by any persistent context
3. **Deleted Entity: An entity is in a deleted (removed) state if**Session.delete(entity) has been called

**JPA Entity Life Cycle Events:** We can perform some events during entity’s life cycle. 2 approaches to perform the events; annotating the methods in entity itself or creating an entity listener class which has call back methods. Void should be the return type.

1. **@PrePersist:** Called before save method is called
2. **@PreRemove:** Called before delete method is called
3. **@PreUpdate:** Called before update method is called, if and only if there is any change in the entity
4. **@PostPersist:** Called before save method is called
5. **@PostRemove:** Called after delete method is called
6. **@PostUpdate:** Called after update method is called
7. **@PostLoad:** Called as soon as loading the entity

Attributes of the ***@Basic*** annotation are applied to **JPA** entities, whereas the attributes of ***@Column*** are applied to the **database** columns. *@Basic* annotation's ***optional*** attribute defines whether the entity field can be *null* or not; on the other hand, *@Column* annotation's ***nullable*** attribute specifies whether the corresponding database column can be *null*

We can use *@Basic* to indicate that a field should be **lazily loaded**

The *@Column* annotation allows us to specify the ***name*** of the mapped database column

**@Size** and **@Length** are same, used for validation of a field.

**@Embedded and @Embeddable ->** Map one entity that contains the embedded properties to a single database table

Hibernate Mapping

**Cascading:** When we perform some actions on target entity, same action will be applied on associated entity

1. **CascadeType.ALL:** All the actions along with hibernate sprcific ones(REPLICATE, SAVE\_UPDATE and LOCK) will be applied on associated entity
2. **CascadeType.PERSIST:** Along with target entity, associated entity also will be saved
3. **CascadeType.MERGE:**

***NOTE:*** Difference between hibernate ***save()*** and ***persist()*** methods is, when we invoke save(), entity will be saved to database irrespective of transaction and flush. When we invoke persist(), entity will be saved in persistence context and changes will be tracked and those changes also will be propagated to database till the transaction.commit or flush is fired