save()(
 => Serializable .save(Object obj)
=> This method gives instructions to save object and also return the assigned or generated identity value back to
the application as the return value.
=> This method is own method of hibernate(not as per specification of JPA).

note: if generators are not configure, then value assigned to id property will be returned as identity value.
eg: increment,sequence,hilo,......

Employee.java
============
```
@Id
@Column(name = "eid")
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Integer empId;
```
As noticed above we have told hibernated to genreate the value of empId, so the generated value is "AutoIncrement"
for MySQLDB

persist()
=>void persist(Object object)
=> return type is void, cannot return the identity value.
=> This method is given by JPA specification and it is implemented by Hibernate.
=> Gives instruction to hiberante to perform save operation on the object.
=> persist() doesnot allows to work with generators.

code
====
```
Employee employee = new Employee();
employee.setEmpId(19);
employee.setEmpName("dravid");
employee.setEmpSal(45444.5);

session.persist(employee);
```
refer: HBPersistOperation

Performing loading operation
=====================
get()[best suited in standalone application]
It perform eager loading.(hits the database and gets the record from dbtable and stores in Entity class object
irrespective of whether we use that Object/not)
if we call get(), automatically the hibernate will generate the sqlquery and hits the database.
even if the record is not available still its the database,as a result of which we say get() is costly in realtime applications

load()[best suited in webapplications]
It perform lazy loading(hits the database only when we use the object data other than primaary keyvalue)
Upong lazy loading,first hibernate creates the proxy object and sets only pk value to it.
when we use getter methods on non-primary keyvalue then hibernate will hit

the database by executing selectquery.
if the record found then it will create a new object and injects the value to that object,otherwise it would result in
"ObjectNotFoundException".

get()
=> It supports eager loading
=> It won't generated proxy object
=> returns null if record not available
=> suitable to check wheter record available or not.
=> Creates only object for Entity class.
=> Best suited for standalone applications(gauranteed that loaded object will be used)
load()
=> It supports lazy loading
=> It generates proxy object
=> It throws ObjectNotFoundException
=> not suitable
=> Creates 2 object(proxy + Entity class)
=> Best suited for webapplications(DAO-> Service-> Controller-> View(jsp using the object is not gauranteed))

Update Operation
===============
1.update()
i> void update(Object object)
This method is used to modify the record of the DBTable.
Set the primary key value and change the other non-primary data for updation.
To use update(), we should remember wheter record exists or not for the give primary key value.
otherwise it would result in "HibernateException".
It would directly generate "update query" without "select query".

ii>Load the object from database and then modify
Here we won't get Exception as the object is available we do modify the Object.

2.saveOrUpdate()
If the object/record is already available only then it will update the record otherwise it will insert/create a new record

3. merge()
On the loaded object, if we want to update the data then we need to go for merge()

4. referesh()
=> will be discussed later to demonstrate the synchronziation of dbrecord to java object.