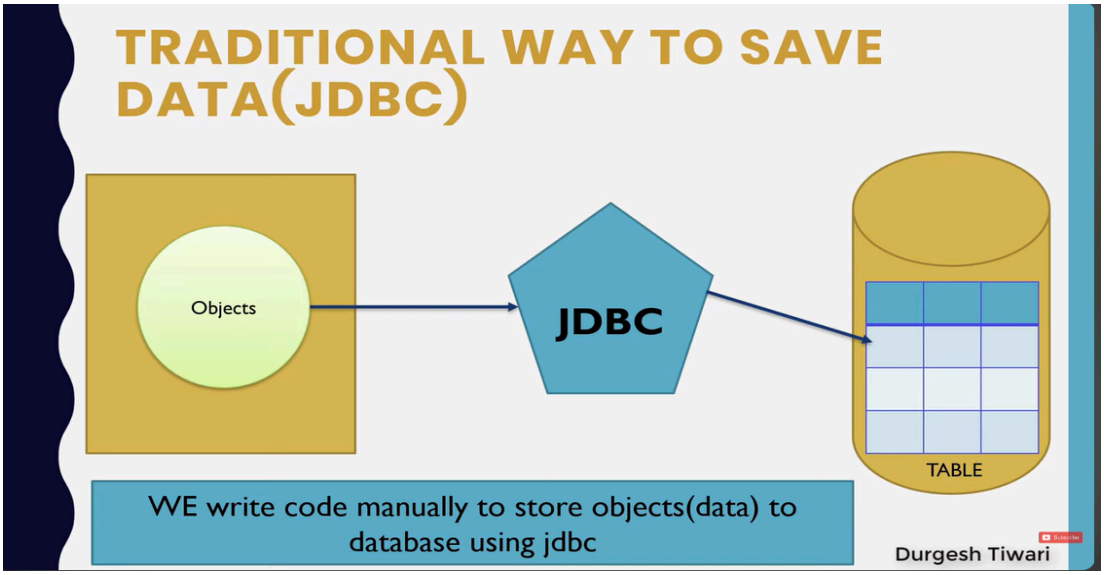
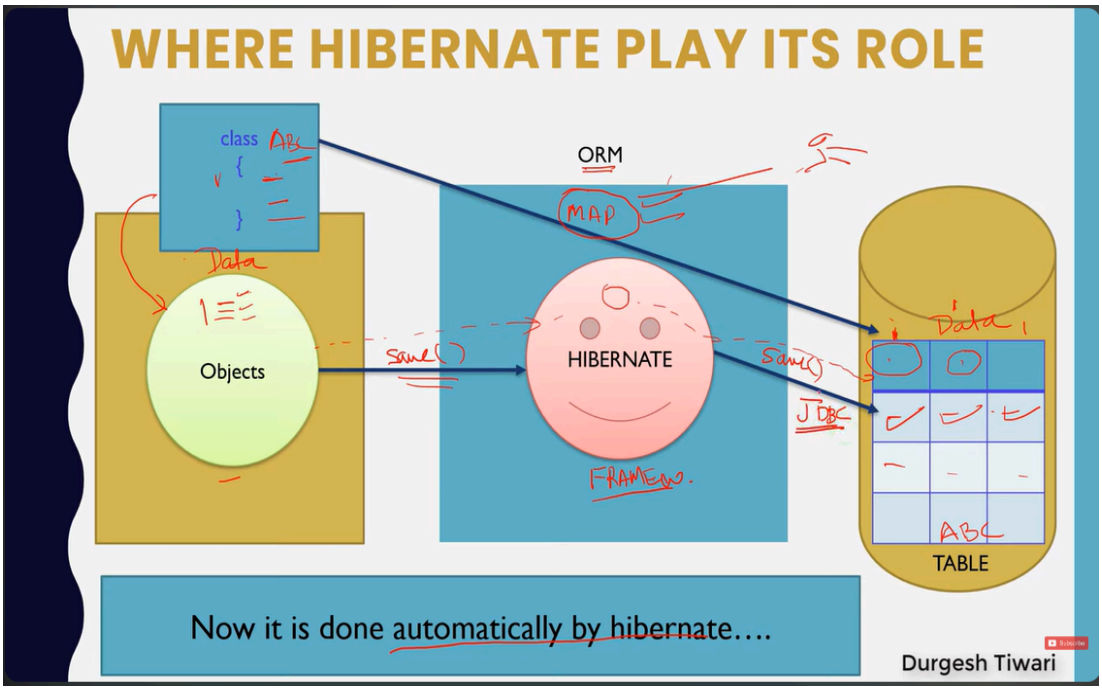
ORM:- Object-relational-mapping

Hibernate is a Java framework that simplifies the development of Java applications to interact with databases.
Hibernate is a non-invasive framework (it doesn't force us to extend or implement any classes).





Connection Factory:-
https://www.digitalocean.com/community/tutorials/hibernate-sessionfactory
Stateless vs stateful session:-
https://www.redhat.com/en/topics/cloud-native-apps/stateful-vs-stateless

Commonly used hibernate annotations:-

1. @Entity:- Used to mark a class as an entity
2. @Table:- Used to change table details
3. @Id:- To mark a column as a primary key (@GeneratedValue:- will automatically inject internal sequence)
4. @Transient:- tells hibernate to not add this variable as a column.
5. @Temporal:- Tells hibernate to save date without time-stamp
6. @Lob:- Tells hibernate this a large object

Fetching data from DB using hibernate

get() vs load()

get( )	load()
get method of Hibernate Session returns null if object is not found in cache as well as on database.	load() method throws ObjectNotFoundException if object is not found on cache as well as on database but never return null.
get() involves database hit if object doesn't exists in Session Cache and returns a fully initialized object which may involve several database call	load method can return proxy in place and only initialize the object or hit the database if any method other than getId() is called on persistent or entity object. This lazy initialization increases the performance.
Use if you are not sure that object exists in db or not	Use if you are sure that object exists.

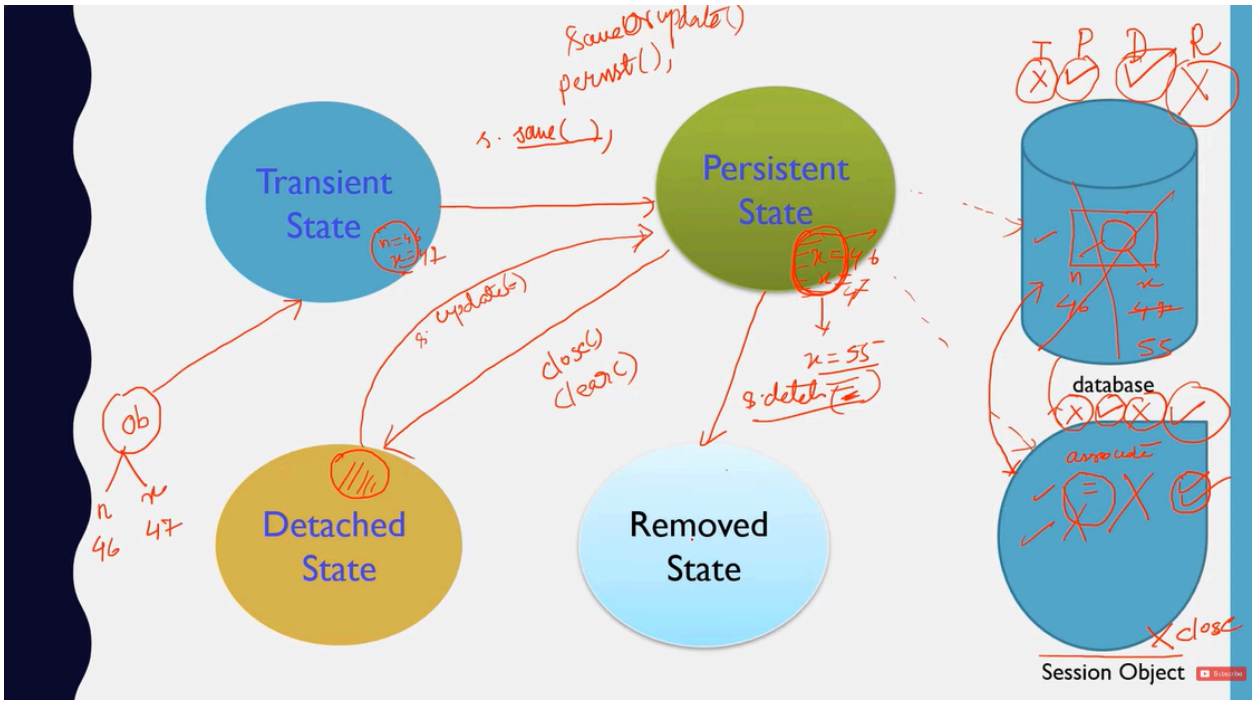
MappedBy signals hibernate that the key to the relationship is on the other side. This means that although you link 2 tables together, only 1 of those tables has a foreign key constraint to the other one. MappedBy allows you to link from the table not containing the constraint to the other table.

There are two fetch types:-

1. Lazy fetch:- Associated data loads only when getter() or size() is called
2. Eager fetch:- Design-pattern in which data loading occurs on the spot

The default is lazy loading.

Hibernate Object states:-



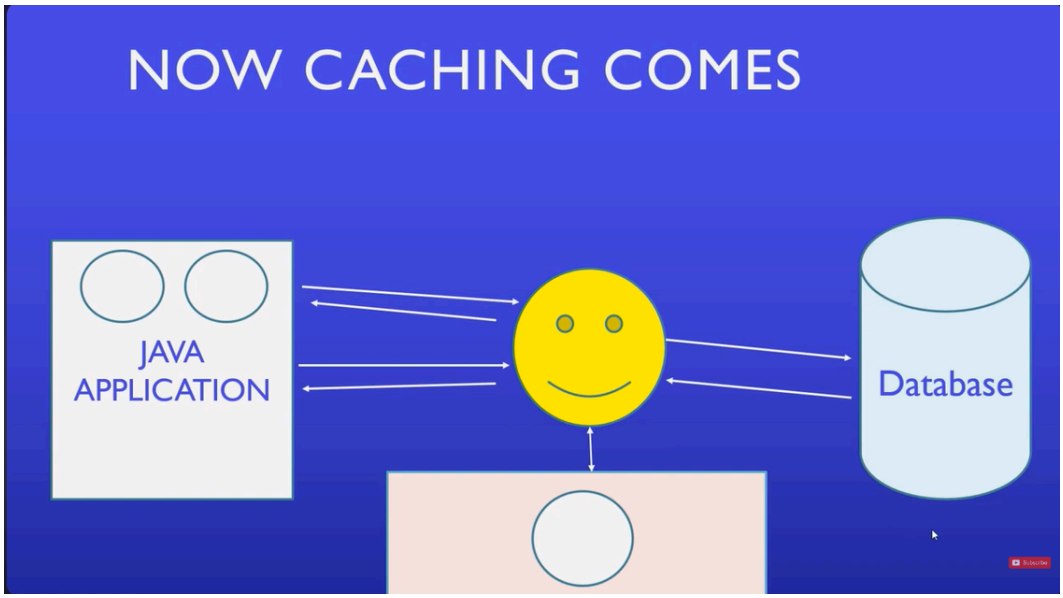

```
student.setId(1414);
student.setName("Peter");
student.setCity("ABC");
student.setCerti(new Certificate("Java Hibernate Course","2 m
//student : Transient

Session s=f.openSession();
Transaction tx=s.beginTransaction();
s.save(student);
//student: Persistent - session,database
student.setName("John");

tx.commit();

s.close();
//student : Detached:
student.setName("Sachin");
```

Caching in hibernate:-
Caching is a mechanism used to enhance the performance of an application. It is used to reduce the number of DB queries fired.



Two types:-
First level:- Associated with session object, it is enabled by default.
Second level:- Associated with sessionFactory object, need to enable it manually.

First level cache demo program:-

```
8
9  public class FirstDemo {
10     public static void main(String[] args) {
11
12         SessionFactory factory = new Configuration().configure().buildSessionFac
13         Session session = factory.openSession();
14         // by default enabled
15
16         Student student = session.get(Student.class, 12424);
17         System.out.println(student);
18
19         System.out.println("working something..");
20
21         Student student1 = session.get(Student.class,12424);
22         System.out.println(student1);
23
24         System.out.println(session.contains(student1));
25
26         session.close();
27
28     }
29 }
```

We can see that the second session.get() doesn't do a DB call.
Once the session is closed and on creating a new session a DB query will be triggered.

Hibernate Criteria

If you don’t like to write any queries this is ideal :-

```
import org.hibernate.Criteria;
import org.hibernate.Session;
import org.hibernate.cfg.Configuration;
import org.hibernate.criterion.Restrictions;

import com.tut.Student;

public class CriteriaExample {
    public static void main(String[] args) {

        Session s = new Configuration().configure().buildSessionFactory().openSession();

        Criteria c=s.createCriteria(Student.class);

        //c.add(Restrictions.eq("certi.course","Android"));
        c.add(Restrictions.like("certi.course", "Java%"));

        List<Student> students=c.list();


        for(Student st:students)
        {
            System.out.println(st);
        }
```