Versioning or ObjectVersioning

It keeps track of how many times object/record is loaded and modified using hibernate.
It generates a special column of type numeric based special number property of Entity class to keep track of the modification.
This special property/col initial value is 0 and it is incrmented by 1 for every modification.
To configure this speical property we need to use one annotation called "@Version".
refer:HBVersioning

TimeStamping

It allows us to keep track of Object is saved(record inserted) and object is lastly updated.
eg: keeping track of when the bank accoutn opened and lastly modified
To do this we use annotations like @CreationTimeStamp,@UpdateTimeStamp
refer:HBTimeStamping

Caching

=>It is a temporoary memory that holds the data for temporary period of time.
=> Cache at client side will hold server data and uses it across the mulitple same requests to reduce the network trip b/w client and server.
=> Hibernate supports 2 levels of Cache
a. First Level Cache(L-1 cache/session cache/default cache)
b. Second Level Cache(L2- cache/configurable cache)

eg: Stockmarket trading,live game score,weather report,......

Note:
session.save(obj),session.saveOrUpdate(obj),sesssion.delete(obj) methods keep the object in L1cache unitll tx.commit() is called.
session.get() will get the object and keep it in L-1 cache and same object will be used across mulitple session.get() method calls with same entity object id.

Caching
a. evict(Object obj) => it will remove particular object from L1-cache
b. clear() -> it will remove all object present in L1-cache.
c. In L1-cache, duplicate objects won't be available.
refer: HBCachingApp

2nd level cache

This caching is associated with "SessionFactory", so we call it as "Global Cache".
Application will start to search for entity object in the following order
a. L1 cache of current session(if not there)
b. L2 cache of SessionFactory object(if not there)
c. Collect from db and keep in L2 cache and L1 cache then give it to application.
It is a configurable cache and we can enbale or disable it.
hibernae supports L2 cache through "EHCache"

```
To configure EHCache in our hibernate projects we use
=============================================
1. Add EHCache jars to the project
2. configure ehcache.xml as shown below
      <ehcache>
            <diskStore path="java.io.tmpdir"/>
            <defaultCache
                         maxElementsInMemory="100"
                  eternal="false"
                  timeToIdleSeconds="10"
                  timeToLiveSeconds="30"
                  overflowToDisk="true"
            />
      </ehcache>
      Also make changes in hibernate.cfg.xml file as shown below
                                    <!-- Configuring EH cache... -->
      <property name="hibernate.cache.use_second_level_cache">true</property>
      <property
name="hibernate.cache.region.factory_class">org.hibernate.cache.ehcache.EhCacheRegi
onFactory</property>
      <property
name="net.sf.ehcache.configurationResourceName">ehcache.xml</property>

3. In the model class inform hiberante to use Caching startegy for Read purpose.
@Entity
@Cache(usage = CacheConcurrencyStrategy.READ_ONLY)//It specifies caching Strategy
public class InsurancePolicy implements Serializable{}

Working with LOB's
===============
To work with LOB in hibernate we use
            @Lob
            private byte[] photo;

            @Lob
            private char[] resume;

                  refer: HBLobOperation


Customgenerator
==============
  Hibernate and JPA had supplied predefined genearator to create primary key value
for almost all databases.
            eg: identity,increment,auto,sequence,.....
if we want a primary key value to be generated for our columns as per our
application needs then we need to go
customgenerators.
      To create our own generator we need to implement an interface called
"IdentifierGenerator"
            It is a functional interface which contains only one method
                  public Serializable generate(SharedSessionContractImplementor
session,Object object) throws HBE

<id name="empId" type="java.lang.Integer" column="eid" >
                  <generator class="in.ineuron.generator.RandomGenerator"/>
</id>
            refer: HBCustomGeneratorApp
```

```
Genearate unique value for student id of iNeuron in the following style
            INEURON0101,INEURON00102,INEURON00103,....
```