**Q1. What is Hibernate?**

* Suppose we have an object of student with variable rollno and name. And we have table with column rollno and name. Now to save the details of student object in table we will first have to create the object set the value, connect to database, write the query and execute it
* ORM help us to directly save the object into database we don’t have to write and execute the queries using JDBC
* Hibernate is an ORM tool that help us to save java object into database table
* We can directly save the object using student.save()method
* Hibernate create the object of session and sessionfactory to establist the connection with database. SessionFactory reads the configurattion file to get the database configuration

**Q2. How do we save an object in hibernate**

* We need the class which we want to save in database with getter and setter methods. The most important thing to remember is the class should be annotated with @Entity above class name and one of variable should be annotated with @Id which will be treated as Primary Key colum. This helps the hibernate to identify table name and class aname
* We will create the object and will set our values
* Now to get started with hibernate we will create the hibernate configuration file”hibernate.cfg.xml” and will have all my database configuration insie this file
* Follow the below steps to save the object in database
* Create the object of Class Configuration
* Create the object of Class ServiceRegistry
* Create the object of sessionfactory and pass the object of registry
* Create the session object
* Create the object of transaction
* Save the object using session.save
* Commit the transaction

**Q3. What is hibernate caching**

* If the same query is running more than one time then there is no need to call the database multiple time when we are using hibernate ORM.
* In order to avoid calling the database for same query Hibernate use the concept of Caching
* There are two levels of caching First Level Cache and Second Level Cache
* First Level cache queries are accessible in the same session.
* If we want to cache the query result and access outside the session object we can use second level cache. Second cache level is implemented by third party Jar file like ecache
* Level One Cache is automatically enabled. IN the below code since the same query is in different session so the query will run twice so the database will also be called two times. In order to access the cache queries outside the session we can use Second Level Cache

Session session = sf.openSession();

Transaction t = session.beginTransaction();

s = (Student) session.get(Student.**class**,112);

s1 = (Student) session.get(Student.**class**,112);

System.***out***.println(s.toString());

System.***out***.println(s1.toString());

t.commit();

session.close();

Session session1 = sf.openSession();

Transaction t1 = session1.beginTransaction();

s1 = (Student) session1.get(Student.**class**,112);

System.***out***.println(s1.toString());

t1.commit();

session.close();

**Q4. How to enable Second Level Cache**

* Add the hibernate-ecanche and net.sf.ecache depency in pom.xml
* We need to enable second level cache and provide Factory class in hibernate configuration file

<property name=*"hibernate.cache.use\_second\_level\_cache"*>true</property>

<property name=*"hibernate.cache.region.factory\_class"*>org.hibernate.cache.ehcache.EhCacheRegionFactory</property>

* The class which is retrieved from the databse will be annotated with
* @Cacheable
* @Cache(usage=CacheConcurrencyStrategy.READ\_WRITE)
* The query can be accessed outside the sessions also

**Q5. What is Hibernate Query Language**

* Till now we were getting the result from session object and using get method.
* What if we want to select particular rows and apply condition on query we can use the Query object from Hibernate
* Query object help us to write the query in the format SQL. And this is called HQL
* We will create a Query object using session.createQuery()
* There is no concept of result set. If we expect a single row from the query then we have to use query.uniqueResult() or if we have multiple rows we have to use query.list() method
* If we want to cache the particular query we have to set query.setCachable(true) for both the query. The first query which is extracting result from database and second query will use result from first query cache. Also I have to enable query cache in hibernate config file

Query q = session.createQuery("from Student where rollno = 11 ");

Student s = (Student) q.uniqueResult();

System.***out***.println(s.toString());

List<Student> sList = q.list();

q = session.createQuery("from Student where rollno < 5 ");

sList = q.list();

**for**(Student s: sList)

System.***out***.println(s.toString());

* Passing an parameter to Hibernate Query language is easy instead of passing harcodre value w have to specify the variable name holding the value preeceding with :b

**int** b = 3;

q= session.createQuery("select rollno,name from Student where rollno < :b ");

q.setParameter("b", b);



**Q6. How can we write HQL Queris where we have to select particular columns?**

* When we specify the column in a HQL Query the return type of result will not be of Class Type that we are extracting the result. The result will be of Object Type.
* If there is one Column then return type will be datatype of column we are extracting. IF we are extratcing column which will have numeric value then return type should be Integer or Long
* If there are multiple column then we have to use Object array

Query q = session.createQuery("select rollno from Student where rollno = 10");

Integer studentObject1 = (Integer) q.uniqueResult();

System.***out***.println(studentObject1);

q= session.createQuery("select rollno,name from Student where rollno = 10 ");

Object[] studentObject = (Object[]) q.uniqueResult();

System.***out***.println(studentObject[0] + " "+studentObject[1] );

**Q7. What is Native SQL Query?**

* We can also write traditional SQL QUeries in place of HQL.
* For this we have to create object of SQLQuery and not the objectof query
* If we want to specific database specific feature in that case we will use Native SQL Query
* Since Native SQL Query does not know for which class the result is getting extracted so we have to add query.addEntity

SQLQuery sql = session.createSQLQuery("select rollno,name from Student where rollno > 10 ");

sql.addEntity(Student.**class**);

List<Student> studList = sql.list();

**for**(Student s : studList)

System.***out***.println(s.getName());

**Q8. What is tthe lifecycle of Hibernate Object?**

* In Hibernate, either we create an object of an entity and save it into the database, or we fetch the data of an entity from the database. Here, each entity is associated with the lifecycle. The entity object passes through the different stages of the lifecycle.
* Folowing are the states of object
  + New(transient)
  + Persistent (when we call sessio.save pr persist method)
  + Detached
* Once we create the object using new keyword the object enters in transient state
* As soon as object is associated with session it goes into persistent state
* Once we call session.detach(object) then object goes in detached state or when the session is closed
* When the object is modifed in persistent state all the changes will be reflected in database

Student sss= new Student();

sss.setName("Hola");

sss.setRollno(8791);

session.save(sss);

sss.setName("Nola");

t.commit();

session.close();

The name in database will be Nola and not Nola since sss object Is still in persistent state

**Q9. Difference between Get and Load method**

* Hibernate Get and load method both help us to derive the data from the database. But the difference is how they are extracting the data
* Load method gives the proxy object. Proxy object is a dummy object which does not have data inside it. The data will be extracted from database when we print the object or use the object in logic

The get method retrieve the data as soon as session.get() method is called and data is available on object whether we use the object or not

Student s = Session.ge(Student.class,101) //data is available

Student sLoad = Session.load(Student.class,101) //data is not available

System.out.prit(sLoad);// data will now be fetched

* The other difference is the exception thrown by Get and Load. If we try to get Student class with roll no 100and student is not available . Get will throw nullPointerException and Load will throw ObectNotFound Exception
* We generally prefer get method in most of scenarios
* Load method is little bit faster than get and Load should be only used when we are confident that record exist is database