**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