1. What is Framework ?

- Framework is an application which is used to develop other applications.
- Framework is a collection of libraries.
- The main purpose of framework is avoid boilerplate coding,which means repetation of the code.
- It is used to develop application in faster mannaer, because it will provide collection of classes & interafaces which will make programmer job easy.
- Ex. In java we have hiberanate framework to work with database application.

2. What is ORM ?

- **ORM** stands for Object Reletional Mapping
- It is a techenique of mapping object oreanted data to releteional data.
- It also map java objects to relational database.
- The main purpose of ORM framework is DATA Persistance.
- It will map java data types with database datatypes,as well as states of an object with columns of the table (States of an object means Non-static variables present in respective object.
- In Java we have 3 types ORM Framework :
    a. Hibernate
    b. Ibatis
    c. Toplink/eclipseLink
- Basically ORM Frameworks are depends on POJO classes ,to make it lightweight (POJO → Plain Old Java Objects)

3. What is DATA Persistance.

- Storing the state of an objects outside the scope of heap area for future access , Is reffered as DATA Persistance.
- Programmer can store states of an object in files like , Exel and CSV as well as Into Database.

4. What Is JPA ?

- JPA stands for java persistence API.
- This is API is similar to javax.persistance packages.
- It dose not have its own implementation.
- It is used to provide specification for ORM Framework.
- Ex. Every ORM framework has different implementation, if programmer want's to make change of hiberanate ORM Framework to Ibatis ORM framework in an application then programmer have to change entire application code.hence developer introduce JPA which will provide common specification for all the ORM Frameworks.
- JPA (Javax.persistance package) consist following classes and interfaces.
    a) EntityManagerFactory interface
    b) Entitymanager interface
    c) EntityTransaction interface
    d) Query interface
    e) Persistance class
- JPA is used to share state of an object from java application to database application.

5. What are the drawbacks of traditional approach while working with database application ?

- Traditional approach means java database connectivity program.
- **Drawbacks of JDBC are as follows :**
  - ❖ It is database vendor dependent because SQL statement syntax will get change from one database vendor to another.
  - ❖ Mannual table creation is manadatory.
  - ❖ Exception handling is manadatory because all the exceptions are checked exceptions like , classNoFoundException , SQLException.
  - ❖ It dose not support cache memory.
  - ❖ Programmer have to perform manual operation on Resultset of database application.
  - ❖ There is a mismatch between object orrentied data representation and relational database data representation.
  - ❖ It leads to boilerplate coding

6. What is hibernate Framewrok ?

- Hibernate is one of the ORM framework .
- Internally it is using , JDBC API and JDBC Driver to communicate with database application.
- It is an open source framework.
- Hibernate is a Non-invasive.
- It is database platform independent.
- Programmer's can execute hibernate program with or withought server.

7. What are the Advantages of Hibernate framework ?

- It is used to store java object into database application.
- It is Database platform independent.

- Ex. If Programmers are developing an application by using ORACLE database vendor and as per client requirement they need to modify it into MYSQL Database vendor then it easy process because of hibernate Framework.
- It generate ifficient query's for java application to communicate with database application.
- Exceptions handling is optional because all the exceptions are unchecked exceptions. Hibernate having translators , to translate checked exceptions into unchecked exception.
- Hibernate supports java persistence query language(JPQL). Which is database platform independent.
- Hibernate have the capability of generating primary key's automatically.
- It is capable to generate tables by using entity classes.
- Hibernate provides first level cache memory ,as well as second level cache memory.
- Hibernate supports HAS-A relationship.
- Hibernate supports mapping between the tables.

8. How to create project hibernate with JPA program ?

1) Step 1 :

- Create a Maven project along with following dependencies.

1. Hibernate core realocation
2. Mysql connector
3. Lombok

2) Step 2 :

- Update the respective Maven Project.

3) Step 3 :

- Create META-INF in src/java/resource folder.

4) Step 4 :

- Create a persistence.xml file in respective folder.
- Go to browser ,serach for persistence.xml file for mysql.
- Go to github repository & copy the code.

9. How dose hibernate intract with database ?

- Programmers are using JAVA Persistance API (JPA) to write code for communicating with hibernate framework.
- Internally hibernate framework will take help of JDBC API & JDBC Driver to communicate with database application.
- In java ,there is only one API to communicate with database application i.e JDBC API
- Hibernate developer has writtern the code to communicate with JDBC API. Where as Programmer's are writing the code to communicate with hibernate framework.

10. What is Entity class ?

- It is used to represent the tables of database application.
- In Entity Class programmers have to create variables which will represent columns of the table.
- Every Entity class must and should get denoted with **@Entity** annotation.
- To represent primary key in given Entity class, programmers are using **@Id** annotation above the variable name.
- Programmers can modify table name by using **@Table** annotation with @Entity.
- Programmers can modify column name as well as can add the constraints by using **@Column** annotation.

- Ex. To add unique constraint , → @Column(unique = true)
- To add NOT NULL constraint , → @Column(nullable = false)
- **Rules to create Entity class.**
  A. Entity class name must be same as table name.
  B. Entity class must not be abstract class.
  C. Each Entity class must and should have atleast one primary key.
  D. Entity class must contain constructor without argument.
  E. Declare all the variables of Entity class as private and provide the access by using public getter and setter methods.
  F. Ex.

```
@Entity
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstrucotr
Public class Student {
        @id
        Private int id;
        Private String name;
        Private String Email;
}
```

11. Explain the use of @Column and @Table annotation.

1) @Column Annotation

- The @Column annotation is used to specify the details of the column to which a field or property will be mapped. We can

use column annotation with the following most commonly used attributes

- name – it represent name of the column to be explicitly specified.
- length - attribute permits the size of the column used to map a value particularly for a String value.
- nullable - attribute permits the column to be marked NOT NULL when the schema is generated.
- unique - attribute permits the column to be marked as containing only unique values.

2) @Table annotation

- The **@Table** annotation allows you to specify the details of the table that will be used to persist the entity in the database.
- The @Table annotation provides four attributes, allowing you to override the name of the table, its catalogue, and its schema, and enforce unique constraints on columns in the table. For now we are using just table name which is EMPLOYEE.
- Ex.

```
@Entity
@Table(name = "EMPLOYEE")
 public class Employee {
   @Id @GeneratedValue
   @Column(name = "id")
   private int id;
 }
```

12. What is EntityManagerFactory ?

- It is as interaface present in javax.persistace package.
- It is develop to perform some specific operation on database like , establishing the connection between java application and database application.
- The main purpose of EntityManagerFactory is , help to create object of EntityManager.
- It will manage multiple entityManagers for the multiple database connections.
- Ex,

EntityManagerFactory factory = Persistence.createEntityManagerFactory("demo");

- In above example createEntityManagerFactory is a static method present in persistence class.
- For above method we have to pass persistence -unit name , specified in persistence.xml file.

13. What is EntityManager ?

- It is an interface , present in **javax.persistace** package.
- It provides multiple methods , to perform operations on database.
- To create **EntityManager** programmers are using **createEntityManager()** method present in **EntityManagerFactory** .
- Ex. EntityManager manager = factory.createEntityManager();
- **EntityManager** consist following methods to perform operation on database **:**
    1. Persist() – To insert data.
    2. Merge() - To update data.
    3. Remove() – To delete data.

4. Find() – To fetch data.
5. createQuery() – To convert string into jpql.

14. What is EntityTransaction ?

- **EntityTransaction** is an interface, present in javax.persistance package.
- It is used to perform transactions on the database like , Insert,update & delete .
- To start the transaction , programmers are using **begin() .**
- To save the transaction programmers are using **commit().**
- A transaction is a set of operations which will either felld or successed as a unit.
- A database transaction consist , set of **DML** operations.
- To get the transaction , programmers are using getTransaction method present in EntityManager.
- Ex.

```
EntityTransaction transaction = manager.getTransaction();
transaction.begin();
transaction.persist(emp);
transaction.commit();
```

15. What is persistence.xml file ?

- It is used to provide database information for EntityManagerFactory.
- It consist multiple tags , to keep database information.
    - <persistece> - It represent collections of peristance-unit.
    - <persistence-unit>
        - represent single database information.

- In single persistence tag programmer can provide multiple persistence-unit tag , to represent multiple database inforamation.
  - <property> - represent a single information about database.

- **Information releted to database are as follows :**
  1. `"javax.persistence.jdbc.driver"` - Used to specify JDBC driver i.e `"com.mysql.cj.jdbc.Driver".`
  2. `"javax.persistence.jdbc.url"` – used to specify url of database i.e . `"jdbc:mysql://localhost:3306/hibernate_a2?createDatabaseIfNotExist=true"`
  3. `"javax.persistence.jdbc.user"` – for username i.e “root”.
  4. `"javax.persistence.jdbc.password"` – for password i.e “12345”.
  5. `"hibernate.dialect"` – dilect property represent type of database and its version. i.e `"org.hibernate.dialect.MySQL8Dialect"`
  6. `"hibernate.show_sql"` – used to display jpql querys on console.
  7. `hibernate.format_sql` – used to format sql querys.
  8. `"hibernate.hbm2ddl.auto"` – used to create table automatically and programmer can provide three values for this property :
     a. Create
     b. Update
     c. Create-drop

16.What is GenerationType ?

- It Is an enum present in javax.persistance package.
- For the table , programmer have to provide primary key data, which has to be unique in nature. So , hibernate have the

capability to generate primary key value automatically with the help of GenerationType.

- It consist major three static and final variable
    1. AUTO
    2. SEQUENCE
    3. IDENTITY
- ***AUTO -***
    - It will generate sequence table , to store next value of primary key and with the help of this table , it will generate sequential values for primary key.
    - For multiple tables programmers will get same sequence table and default name of this sequence table is hibernate_sequence.
    - For different database different sequence table will get generate.
- ***Identity –***
    - It generate primary key value based on database.
    - Always it starts from 1.
- ***SEQUENCE –***
    - It is same as AUTO, but programmer create custmize sequence.

- To specify generation type “make use of @Generated value annotation with @Id annotation “
- Ex.

```
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
```

17. What is cascadeType ?

- It is used to apply modifications of current entity to map entity.

- CascadeType is an enum, it contains 4 static and final variables are as follows :
  I. PERSIST
  II. MERGE
  III. REMOVE
  IV. ALL
- Ex. If programmer wants to save employee along with salary details then salary account deatials should get save automatically. It is possible by using cascadeType.
  @OneToOne(cascade = cascadeType.PERSIST)
- **PERSIST** – use for saving the data.
- **MERGE** – used to update the data.
- **REMOVE** – used to delete the data.
- **ALL** – used to perform insert, update ,delete operations.
- CascadeType is basically used for data manipulation language.

18. What is FetchType ?

- FetchType Is an enum.
- Generally all the mappings are assigned with fetchType.
- Ex. Bydefault fetchType of all the mappings are as follows :
  a) ONE-TO-ONE → EAGER
  b) ONE-TO-MANY →LAZY
  c) MANY-TO-ONE →EAGER
  d) MANY-TO-MANY →LAZY
- FetchType consist two static enums ,
- ***LAZY*** – if fetch type is LAZY then data will get fetch only from current entity.
- ***EAGER*** – If fetch type is EAGER then data will get fetch current entity as well as map entity.

- Ex . if programmer is trying to fetch data from employee then he will get the data of employee as well as salary account table.

@OneToOne (fetch = FetchType.LAZY)

19. What is JPQL ?

- It is stands for java persistence query language.
- It is used to perform operations on database application irrespsctive with primary key column.
- It is a platform independent query language.
- In JPA we have find () , merge() ,remove () which will work respective with primary key column.hence to overcome from the drawbacks of above methods , programmers are using jpql.
- JPQL does not supports insert querys.
- In SQL querys programmers are using table name and column name, whereas in jpql programmers are using className and variablesName.
- Ex.

  String delete = "DELETE FROM Emp e1 where e1.mobile = 98533";
- To convert String into jpql programmers are using createQuery () present in Entitymanager interface.
- The return type of this method is Query interface.
- Ex .

  Query query = manager.createQuery(delete);
- **Query interface consist four methods are as follows :**
  A. **executeUpdate()**
    - used to execute update and delete querys,return type of this method is int.

- Ex.

```
Int status = query.executeUpdate();
```

B. **getSignleResult()**

- used to execute select query which will return only one record.
- This methods return type is object class , hence programmer have to perform downcasting.
- If given data is not present in database then it throws an exception “NoResultException”.
- Ex.

```
String select = “SELECT  e1 FROM Emp e1 WHERE e1.email = “kg@gmail.com “ ;

Query jpql = manager.createQuery(select);
try{
        Emp e1 = (Emp) jpql.getSingleResult();
        System.out.println(Emp);

}catch(NoResultException e){
    System.err.println(“Invalid email id “ );
}
```

C. **getResultList ()**

- used ot execute the select query which return multiple records.
- The return type of this method is List.
- Ex.

```
String select = “FROM Emp e1 WHERE e1.sal >= 213455”;

Query jpql = manager.createQuery(select);

List<Emp> emps = jpql.getResultList();

System.out.prinltn(emps.isEmpty() ? “No data found“:emps);
```

D. **setParameter ()**

- used to allocate values for placeholders & parameters.

20. How to work with runtimes values in jpql ?

- We have two approaches to provide runtimes values in jpql are as follws :
  1. Placeholder 2. Parameter.
- Ex.

1. Placeholder –

String query = “From Emp e1 where e1.ename = ?1” ;

2. Parameter -

String query = From Emp e1 where e1.ename = :en”;

- To allocate value for this placeholder and parameter programmers are using setParameter() present in Query interface.
- Ex.
  a) Placeholder – jpql.setParameter(1 ,”xyz”);
  b) Parameter = jpql.setParameter(“en” , “xyz”);
- NOTE –
  - while wrttining select query programmers cant use astrics (*) to select all the columns
  - **Few Ex. Of select querys :**

1) Select e1 from emp e1 → valid
2) Select * from emp e1 → invalid
3) From emp e1 → valid
4) Select e1 * form emp e1 → invalid
5) Select e1 from emp → invalid
6) From emp →invalid
7) Select e1 from emp e1 where e1.empId = 101 → valid

21. What is the difference between SQL and JPQL ?

SQL	JPQL
Programmers are using tableName and columnName to write querys.	Programmers are using className and variableName to write query.

We have different setters methods for each data type, to allocate values for placeholder	We have only one method i.e setParameter to allocate values for placeholeders and parameters.
Select clause is manadatory.	Select clause is optional.
To select all the columns programmers are using astic (*).	To select all the columns programmers are using alieas name.
It is database platform dependent .	It is a database platform independent.

22. What is hibernate mapping ?

- It is used to create reletion between more than one table/classes.
- Basically , it used to create foreign key into the tables.
- To archieve hibernate mapping , programmers are using HAS-A relationship which means one object is depends on another object.
- We have 4 types of hibernate mapping.
  1) ONE-TO-ONE
  2) ONE-TO-MANY
  3) MANY-TO-ONE
  4) MANY-TO-MANY
- We can archieve haibernate mapping in two ways :
  a. Unidirectional mapping
  b. Bidirectional mapping
- Ex .
  1) If book has author and programmers is trying to fetch the data from book along with author and programmers is

trying to fetch data form author along with book is refferd as bidirectional mapping.

2) Programmer can fetch author details along with bookdetails but programmer can fetch bookdetails along with is Author deatials refferd as unidirectional mapping.

- Ex for ONE-TO-ONE Unidirectional mapping.

```
//Child table
@Entity
Public class Person{
    @Id
    Private int id;
    Private String name;

    @OneToOne
    Private Passport passport
}
//Parent table
@Entity
Public class Passport{
    @Id
    Private int id;
    Private Localdate issuedDate;
}
```

- Ex. One-to-Many and Many-to-One Bidirectional mapping.

```
@Entity
@Getter
@Setter
@NoArgConstructor
@allArgsConstructor
```

```
Public class Customer{
    @Id
    Private int id;
    Private String name;
    @OneToMay(mappedBy = “customer”)
    Private List<Expense> expenses;
}

 @Entity
@Getter
@Setter
@NoArgConstructor
@allArgsConstructor
Public class Expense{
    @Id
    Private int expenseId;
    Private double amount;

    //cant used mappedBy
    @ManytoOne
    Private Customer customer
}
```

23. What is the use of mapped by attribute ?

- Bydefault , bidirectional mapping will create duplicate foreign keys into releted table. To avoid this duplicate foreign keys programmers prefers mappedBy attribute with @OneToOne, @OneToMany and @ManyToMany annotations.
- Programmer cant use mappedBy with @ManyToOne annotation.

- Always specify mapped by in parent class.

24. Explain usage of @JoinColumn and @Jointable?

- **@JoinTable** stores the id of both the entity into a separate table wheres **@JoinColumn** stores id of the another entity in a new column in same table.
- Use **@JoinColumn** when the entities have direct relationship i.e a foreign key between two entity.
- Use **@JoinTable** when you manage the relationship between entities in another table.
- **@JoinTable** : Use this when you need a more normalized database. ie. to reduce redundancy.
- **@JoinColumn** : Use this for better performance as it does not need to join extra table.
-

25. Explain Usage of @SequenceGenerator ?

- Sequence generator is generates numeric sequence values such as 1, 2, 3, and so on.
- It does not affect the number of input rows. The Sequence Generator transformation is used to create unique primary key values and replace missing primary keys.

@SequenceGenerator ➔TO provide customize sequence.

- Ex.

```
@SequenceGenerator(name = "book_id" , initialValue = 100 ,
allocationSize = 4,sequenceName="Hiberanate_sequence_size")
```

Generator → to specify sequence_generator name for generetedValue

name → to specify sequence_generator

initialvalue → to specify starting value for sequence

allcationSize → to specify increment size of sequence

sequenceName → to provide sequence table name

26. Write a difference between hibernate with JPA and JDBC ?

JDBC	Hiebrnate
JDBC is a technology.	Hibernate is ORM Framewrok.
It dosent supports the cache memory	It supports the cache memory.
It is database vendor dependent	It is database platform independent.
Exception handling is mandatory.	Exception handling is optional.

27. What is cache Memory in hibernate ?

- Cache memory is a temprory storage area.
- Caching helps to improve application performance.
- Idea of using caching to reduced the number of database querys.
- Hibernate provides two types of cache memory :
    i. First-level-cache
    ii. Second-level-cahche
- Bydefault first-level-cache is enable in hibernate.

- Bydefault , hibernate dose not provide second level cache , hence programmer have to use third party libraries to archive second level cache.
- Ex for Third party libraries, Ehcache, oscahce, jbosscache.
- First-level-caching is depend on EntityManager. Whereas second-level-cache is depend on EntityManagerFactory.
- For each and every EnityManager there will be separate cache memory.
- For each and every EntityManagerFactory there will be separate cache momory.
- In one EntityManagerFactory, programmer can create multiple EntityManager.
- When programmers send request then firstly record will check in first level cache, if it is not present then it will get chach into second-level-cache, if it is not there then it will get access from database.

28. How to enable second level cache ?

a. Step 1

- Add third party libraries by using hibernate ehcache dependency.
- In persistence.xml file make use of following tags,

```
<shared-cache-mode>ENABLE_SELECTIVE</shared-cache-mode>

<properties>

<property name="hibernate.cache.use_second_level_cache" value="true"/>

<property name="hibernate.cache.region.factory_class"
value="org.hibernate.cache.ehcache.internal.EhcacheRegionFactory"/>

<properties>
```

- Denote Entity class by using @Cacheable annotation.

29. Explain hibernate life cycle ?

- There are four phases in hibernate life cycle.
  i. Transient :
  ii. Persistent :
  iii. Detach :
  iv. Remove :
- When programmer creates an Entity class object , then object will be in transient state.
- When programmer call persist method & merge method then object will be In persistent state.
- When programmer call detach method then object will be In detach state.
- When programmer calls remove method then object will be in removed state.
- All the operations perform on specific object , will be eligible for garbage collector.

30.

31.

32.

33.