1. **what is framework ?**

- Framework is an application which is used to develop other application.
- Framework is a collection of libraries.
- The main purpose of framework is avoiding boiler plate coding which means repetition of the code.
- It is used to develop application in faster manner, because it will provide collection of classes and interfaces which will make programmer job easy
- For ex :- In java we have hibernate framework to work with db. application.

2. **What is ORM ?**

- ORM stand for object relational mapping.
- It is a technique of mapping object-oriented data to relational data.
- It also mapped java objects to relational db.
- The main purpose of ORM framework is data persistence.
- It will map java datatype with database datatype as well as states of an objects with columns of the table (states of an objects :- means non-static variables present in respective object)
- In java we have 3types of ORM framework.
    - **Hibernate**
    - **ibatis**
    - **Toplics**
- Basically, ORM frameworks are depends on POJO classes.to make it ,Lightweight (POJO stands for plane old java objects).

Java Entity ----------> ORM tool ----------> RDBMS

java objects SQL statement

3. What is data persistence ?

- Storing the states of an object outside the scope of heap area for future access is referred as data persistence
- Programmer can store states of an objects in files like excel and csv as well as into DB.

4. What is JPA ?

- JPA stands for java persistence API.
- This API is similar to javax.persistence package.
- It does not have its own implementation.
- It is used to provide specification for ORM framework.
- For ex :- Every ORM framework has different implementation, if programmer wants to make change of hibernate ORM framework to ibatis ORM framework in an application the
- programmer have to change entire application code; hence developer introduce JPA which will provide common specification for all the ORM frameworks.
- JPA (javax.persistence package) consist following classes and interface
    - **EntityManagerFactory Interface**
    - **EntityManager Interface**
    - **EntityTransaction Interface**

- **Query Interface**
- **Persistence Class**

- JPA is used to share states of an objects from java to db application.

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

- Traditional approach means java database connectivity program.
- Drawbacks of JDBC are as follows
  - It is database vender dependent because SQL statement syntax will get change from one db vender to another
  - Manual table creation is mandatory
  - Exception handling is mandatory because all the exceptions are checked exceptions like ClassNotFound Exception and SQL exception.
  - It does not support cache memory
  - Programmer how to perform manual operation on ResultSet of database application.
  - There is a mismatch between object-oriented data representation and relational database data representation.
  - It leads to boiler plate coding.

6. What is hibernate framework ?

- 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 framework.
- It is database platform independent.
- Programmer can execute hibernate program with or without server.

7. What are the advantage of hibernate framework?

- It is used to store java objects into database application.
- It is database platform dependent.
- E.g.: If programmer are developing an application by using oracle database vender and as per the client requirement they to modified into my SQL database vender then it is easy process because of hibernate framework.
- It generates efficient queries for java application to communicate with database application
- Exception handling is optional because all the exceptions are unchecked exception. hibernate having translator, to translate checked into unchecked exception.
- Hibernate supports java persistence query language(JPQL) which is database platform independent.
- Hibernate have the capability of generating primary keys automatically.
- It is capable to generate tables by using entity classes.
- Hibernate provide 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 for hibernate with JPA programs

Step-1 Create a maven project along with following dependencies

- **Hibernate core relocation.**
- **My SQL connector**
- **Lombok**

Step-2 Update the respective maven project.

Step-3 Create META-INF folder in source/java/resource folder.

Step-4 Create a persistence.xml file in respective folder.

Step-5 Go to browser and search for persistence.xml file for MySQL.

Step-6 Go to GitHub repository and copy the code.

9. How does hibernate interact database.

- Programmers are using java persistence API, to write code for communicating with hibernate framework.
- Internally hibernate framework will take help of JDBC Api and 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 written the code to communicate with JDBC Api whereas programmers are writing the code to communicate with hibernate framework.

10. What is Entity class ?

- It is used to represent tables of database application.
- In entity class programmer have to create variables which will represent columns of the table.
- Every entity class must and shoot 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 tables name by using **@Table** annotation where **@Id.**
- Programmer can modify column name as well as can add the constraint by using **@Column** annotation.
- for ex-
    - To add unique constraint **@Column(unique=true)**
    - To add NotNull constraint **@Column(nullable=false)**
- **Rules to create entity class**
    - Entity class name must be same as table name.
    - Entity class must not be abstract class.
    - Each entity class must have at least one primary key.
    - Entity class should contain constructor without argument.
    - Declared all the variables of entity class as private and provide the access by using public getter and setter methods
    - For 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 represents 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 entity manager factory ?

- It is an interface present in javax.peristence package.
- It is developed to perform some specific operations on database like establishing the connection between java and database application
- The main purpose of entity manager factory is, help to create object of entity manager.
- It will manage multiple entity managers for the multiple database connections.
- For ex - EntityManagerFactory factory = Persistence.createEntityManagerFactory("demo");
  In above ex- Create entity manager factory is a static method, present in persistence class.
- For above method we have to pass persistence unit name, specify in persistence.xml file.

13. What is entity manager ?

- It is an interface, present in javax.persistence package.it provides multiple methods, to perform operations on database
- To create entity manager, programmers are using (createEntityManager()) present in entity manager factory.
- For ex- EntityManager manager = factory.createEntityManager();
- Entity manager consist following methods, to perform operations on database.
    - persist()-To insert data
    - Merge()-To update data
    - Remove()-To delete data
    - Find()-To fetch data
    - CreateQuery()-To convert string into JPQL query.

14. What is entity transaction ?

- Entity transaction is an interface, present in javax.persistence 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 transactions programmers are using **commit().**
- A transaction is a set of operations which will either frame or succussed as a unit.
- A database transaction consist, set of DML operations.
- To get the transaction, programmers are using get transaction() present in entity manager.
- For 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 entity manager factory
- It consist multiple tags, to keep database information.
- <persistence> - It represents collection of persistence unit.
- <persistence-unit> - Represents One database information.
- note-In single persistence tags, programmer can provide multiple persistence unit tag, to represent multiple database information.
- <property> - Represent a single information above database.

Information related to database are as follows.

- *"javax.persistence.jdbc.driver"* - used to specify jdbc driver i.e. com.mysql.cj.jdbc.Driver
- *"javax.persistence.jdbc.url"*- used to specify url of db
- *"javax.persistence.jdbc.user"*- for username i.e. “root”.
- *"javax.persistence.jdbc.password"*- for password i.e. “12345”.
- *"hibernate.dialect"* - dialect property represents type of database and its version
- *"hibernate.show_sql"*- used to display JPQL queries on console
- *"hibernate.format_sql"*- used to format SQL queries
- *"hibernate.hbm2ddl.auto"*- used to create table automatically and programmer can provide three values for this property
    - create
    - update

- create_drop

16. What is generation type ?

- It is an Enum, present in javax.persistence 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 generation type
- It consists major three static and final variables
    - AUTO
    - SEQUENCE
    - IDENTITY

AUTO

- Auto will generate sequence table to store next value of primary key and with the help of these table, it will generate sequential values for primary key.
- For multiple tables, programmer will get same sequence table and default name of these sequence table is hibernate_sequence.
- For different database different sequence table is generate.

IDENTITY -

- It generates primary key value based on database.
- Always it start from 1

SEQUENCE -

- It is same as auto, but programmer can create customize sequence
- To specify generation type "make use @generatedvalue annotation with @id annotation"

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private int id;
```

17. What is cascade type ?

- It is used to apply modification of current entity to mapped entity
- Cascade type is an Enum it contains 4 static and final variables are as follows
    - **PERSIST**
    - **MERGE**
    - **REMOVE**
    - **ALL**
- For ex - if programmer wants to save employee along with salary details then salary account details should get save automatically. it is possible by using cascade type.

```
@oneToone(cascade = cascadeType.PERSIST)
```

- **PERSIST** is used to save the data.
- **MERGE** is used to update the data.
- **REMOVE** is used to delete the data.
- **ALL** is used to perform insert, update and delete operations.
- Cascade type basically used for Data manipulation statement.

18. What is fetch type ?

- Fetch type is an Enum
- Generally, all the mappings are assigned with fetch type. for ex- by default fetch type of all the mappings are as follows

OneToOne → Eager

OneToMany → Lazy

ManyToOne → Eager

ManyToMany→ Lazy

- Fetch type Enum consists to static and final variables.
- **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 from current entity as well as mapped entity.
- **For ex** - If programmer is trying to fetch the data from employee then we 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 irrespective 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 drawback of above method, programmers are using JPQL.
- JPQL does not supports insert queries.
- In SQL queries programmers are using table name and column name where as in JPQL programmers are using class Name and variables name.
- for ex - String delete = "delete from employee e1 where e1.mobile = 987897898";
- To convert string into JPQL, programmers are using create query method present in entity manager interface
- The return type of method is query interface.
- For ex-

```
Query query = manager.createQuery();
```

- **Query interface consists 4 methods are as follows -**

1) executeUpdate –

- Used to execute update and delete queries, return type of method is int.
- For
- ex-

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

2) getsingleresult() –

- Used to execute select query which will return only one record.
- this method is return type is object class hence programmer have to perform down casting.
- if given data is not present in a database then it throws an exception "NoResultException".

- for 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 “ );
}
```

3) GetResultList -

- Used to execute the select query which returns multiple records. the return type is List.
- for 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);
```

4) setParameter() –

- Used to allocate values for placeholder and parameters.

20. How to work with runtime values in JPQL ?

- We have approaches, to provide runtime values in JPQL are as follows
  - By using placeholders

```
String query = "from employee e1 where e1.ename = ?1";
```

  - By using parameters

```
String query = "from employee e1 where e1.ename = :en";
```

- To allocate the values for this placeholder and parameter programmers using set parameter method present in query interface.
- For ex –
  - Placeholder → jpql.setParameter(1,"xyz");
  - Parameter → jpql.setParameter(“en”,"xyz");
- Note
  - While writing select query programmers can’t use * to select all the columns.
  - few examples of select query----
    - Select * from employee e1 --> invalid
    - Select e1 * from employee e1 --> invalid
    - Select e1 from employee e1 --> valid
    - Select e1 from employee --> invalid
    - Select e1 from employee where e1.empId = 101 -->valid

21. What is difference between SQL and JPQL

SQL	JPQL
Programmers are using table name and column name to write SQL queries	Programmers are using class name and variable name to write the queries
We have different setter method for each datatype, to allocate values for placeholder	We have only one method i.e. setParameter to allocate values for placeholder and parameters
Select clause is mandatory	Select clause is optional
To select all the columns programmers are using *	To select all the columns programmers are using alias name
It is a database platform dependent	It is a database platform independent

22. What is hibernate mapping ?

- It is used to create relation between more than one table or classes.
- Basically, it is used to create foreign key into the tables
- To achieve hibernate mapping programmers are using has-a-relationship which means one object is depends on another object.
- We have 4types of hibernate mapping :
    - OneToOne.
    - OneToMany
    - ManyToOne
    - ManyToMany
- We can achieve hibernate mapping in two ways :
    - unidirectional
    - Bidirectional
- for ex –
    - If book has an author and programmer is trying to fetch the data from book along with author and programmer is trying to fetch the data from author along with book is referred as bidirectional mapping.
    - Programmer can fetch author details along with book details but programmer can't fetch book details along with author details is referred as unidirectional mapping.

- Example for **OneToOne unidirectional mapping** -

```
@Entity
public class Passport {
        @Id
        private int id;
        private LocalDate issuedDate;
}
@Entity
public class Person {
        @Id
        private int id;
        private String name;

        @OneToOne
private Passport passport;
}
```

- **Example for oneTomany Bidirectional mapping**

```
@Entity
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class Customer {
	//Parent
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int id;
	private String name;
	private long mobile;

	@OneToMany(mappedBy = "customer")
	private List<Expense> expense;

@Entity
@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
public class Expense {
	//child
	@Id
	@GeneratedValue(strategy = GenerationType.IDENTITY)
	private int id;
	private double amount;
	private String category;

	@ManyToOne()
	private Customer customer;
}
```

23. What is used of mappedBy attribute ?

- By default, bidirectional mapping will create duplicate foreign keys into related table. to avoid these duplicate foreign keys,
- programmers preferred mappedBy attribute with **@OneToOne , @OneToMany and @ManyToMany** annotation.
- Programmer can't use mapped by with **@ManyToOne** annotation.
- Always specify mapped by in parent class.

24. Explain usage of @join column and @join table annotation

- **@JoinTable** stores the id of both the entity into a separate table whereas **@JoinColumn** stores id of another entity in a new column in same table.
- Use **@JoinColumn** when the entities have direct relationship i.e. a foreign key between two entities.
- Use **@JoinTable** when you manage the relationship between entities in another table.
- **@JoinTable :** Use this when you need a more normalized database. i.e. 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 generating 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.
- **E.g.:**

```
@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
- **allocationSize** →to specify increment size of sequence
- **sequenceName** →to provide sequence table name

26. Write a difference between hibernate with JPA and JDBC

JDBC	Hibernate
JDBC is a technology.	Hibernate is ORM Framework.
It doesn't support 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 temporary storage area.
- Caching helps to improve application performance.
- Idea of using caching is to reduce the number of database queries.
- Hibernate provides two types of cache memory
    - First_level_caching
    - Second_level_caching
- By default, first level caching is enabled in hibernate.
- By default, hibernate does not provide second level caching hence programmer have to used third party libraries to achieve second level caching.
- Example for third party libraries
    - Ehcache
    - Oscache
    - Jbosscache
- First level caching is depending on entity manager whereas second level caching is depending on entity manager factory.
- For each and every entity manager, there will be separate cache memory.
- For each and every entity manager factory , there will be separate cache memory.
- **note** In one entity manager factory, programmer can create multiple entity manager.
- When programmer sends the request then firstly record will get check in first level cache, if it is not present the it will get check into second level cache, if it is not there then it will get accessed from database

28. How to enable second level cache ?

- **Step1.**At third party libraries by using hibernate ehcache dependencies
- **Step2.**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>
```

- **Step3.**Denote entity classes by using **@cacheble** annotation

29. Explain hibernate lifecycle?

- There are four phases in hibernate life cycle.
    - **Transient :**
    - **Persistent :**
    - **Detach :**
    - **Remove :**
- When programmer creates an Entity class object , then object will be in transient state.
- When programmer call persists 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.