Hibernate Interview Question

1) What are the Drawbacks of JDBC?

- In all the cases, all the steps will remain the same except the 4th step; this condition is called **Duplication of code** (or) **Boiler Plate Code.**
- To avoid this **"Duplication of code** (or) **Boiler Plate Code"**, we use **SingleTon Class.**
- Where **SingleTon** class always returns 2 methods
  - a) for Connection ⇒ Trying to establish a connection with the DB server
  - b) closing the Connection ⇒ To close the Connection [since it is a costly resource]

✸ **JDBC is always dependent on Database.**

[Since the SQL queries varies from one Database Server to another.]

- Hibernate is independent of Database, since hibernate automatically generates SQL queries as per the Database

✸ **JDBC does not support auto-generation of Tables & Primary Key.**

- In case of hibernate, hibernate will automatically generate the tables on its own along with assigning primary key.

✸ **JDBC does not support "cache mechanism"**

"**Cache memory**":- It is a temporary (or) buffer area, which stores constant data / repeated data from the Database.

(**) cache mechanism avoids traffic between Java Application and Database Server, hence performance of an application increases.

2. What is ORM and mention the tools and specifications of ORM?

```
public class Student {
      public int id;
      public String name;
      public double perc;

public static void main(String arg[]){
      Student s=new Student();
      s.id=1;
      s.name="Uday";
      s.perc=69.2;
  }
}
```

Class ⇒ Class
Variables ⇒ Variables
Object ⇒ Object

Object Model

Table ⇐ Student

id	name	perc
1	Uday	69.2

Columns ⇐ id, name, perc
Record/row ⇐ 1, Uday, 69.2

Data Model Or Relational Model

ORM

Class	=	Table
Variables	=	Columns
Object	=	Record/row

Object Model

Why it is called as Object Model?

Until we create an Object we cannot access one single functionality

Data Model (or) Relational Model

Why it is called as Data Model?

Apart from data we don't find any other components

"Conversion of Java Object into Relational Model is known as Object Relational Mapping".

Or

"The Object which created in case of Java must mandatorily have a relationship with data's in the Database server is known as **Object Relational Mapping** [**ORM**]".

- There are different types of mechanism / tools present in ORM namely;
  i. Hibernate
  ii. TopLink
  iii. Ibatis etc...

❖ **Specifications of ORM:-**

There are 3 different specifications wrt ORM;

i. Every Java Bean Class / Entity Class / POJO class represents a Table.
ii. Every Variable of Java Bean Class represents a Column.
iii. Every Java Bean Class Object represents a Record/row.

ORM		
Class	=	Table
Variables	=	Columns
Object	=	Record/row

3. Define Framework and explain the categories of Framework?

- Framework is a set of API's (or) semi-software (pre-written code) which is used to develop an application in a loosely coupled manner.
  Ex.:- Ready-made template
- It is used to develop an application in a simplified manner (or) using which we can develop an application in a simplified manner.
- There are 2 different categories present in framework;
  i. Invasive framework
  ii. Non-invasive framework

i. **Invasive Framework:-**

"***Framework which allows to*** extends one of their classes / implements one of their interfaces".

Ex.:- Struts, EJB framework

ii. **Non-invasive Framework:-**

"***Framework which does not allow to*** extend one of their classes / implements one of their interfaces".

Ex.:- Hibernate and Springs

4. Difference between API and Framework?

(***)Difference between API and Framework

:API:	:FRAMEWORK:
Is an interface between two different application	Is used to design / develop application such as MVC web application
Ex.: Java-Collection	Ex: Java-Array

5. Define Hibernate?

"Hibernate is an open-source, non-invasive framework, ORM tool which is used to convert Java Model [classes & variables] into Relational Model [Tables & Columns]"

6**. What are the advantages of Hibernate?**

a. Hibernate is independent of Database.
   [Since SQL queries are automatically generated by hibernate as per the Database]
b. Hibernate supports auto-generation of Tables and Primary key.
c. Hibernate supports cache mechanism (avoids traffic between java application and Database server).
d. Hibernate supports Connection Pooling (resources which stores Database connection)
   [i.e., from one single java application we can hit multiple Database servers at a time]
e. Hibernate supports HQL [Hibernate Query Language]
f. (***) Hibernate supports dialect

7. Define the properties of Dialect?

Dialect: - **i)** It is responsible for generating SQL queries based on each Database

ii) To achieve Object Relational Mapping.

8. What are the pre-requisites for Hibernate Java Application?

a. Open Java Perspective and select *Navigator* mode
b. Right click within Navigator mode and create a *new* Java Project & name it
c. Right click on Project and create a new folder called lib and add all the Jar files into the lib folder and build a java path to import the properties from the jar file.
d. Add the configuration file [hibernate.cfg.xml] into the source folder
e. Select src folder and create a new package structure
f. Select application name and create a class [java bean class / entity class / POJO class]

9. Explain the different JPA Annotations?

@Entity	specifies it is an Entity class which is used to write Hibernate logic
@Table	specifies Table name in the Database which maps Entity [Table name will be associated with Entity class]
@Id	specifies it is Primary Key where the data type must be int or long
@GeneratedValue	specifies auto-generation of Primary Key

10. What is the use of DAO class?

DAO class **Data Access Object**

Why we use DAO class?

- Since DAO class is the only area where we can access the properties or functionalities of Java Bean Class / Entity Class / POJO Class
- Only class / area to execute java application which has main method in it. [JVM is responsible for execution]

11. Difference between get( ) and load( ) ?

get( )	**load( )**
it directly hits the DB server and returns the real object/actual object	It always returns a Proxy object without hitting the DB server
If the ID is not present in the table, then	If the ID is not present in the table, then
get( ) throws NullPointerException (or) NULL	load( ) throws ObjectNotFoundException

12. Define Proxy Object?

Proxy object is an Object with a given ***Id*** but the properties are not initialized yet

13. What are the steps to be followed in DAO class?

a. Create an object of **Java Bean Class** / **Entity class** / **POJO class**
b. Set the value of the members of Java Bean Class or Entity class
c. Create an object of Configuration class present in org.hibernate.cfg package
***Syntax*:**

```
Configuration conf =new Configuration();
```

d. Call a zero-parameterised **`configure()`** method which is declared in Configuration Class.
***Syntax*:** `conf =configure();`

> Why to call a Zero-parameterised **`configure()`** ?
> **`configure()`** is responsible for loading and validating the configuration file called hibernate.cfg.xml

e. Create an implementation object of **SessionFactory** *interface* present in org.hibernate package by using a factory / helper method called *buildSessionFactory( )* which is declared inside Configuration class
***Syntax*:**

```
SessionFactory sef=conf.buildSessionFactory();
```

f. Create an implementation object of **Session** *interface* present in org.hibernate package by using a factory / helper method called *openSession( )* which is declared inside **SessionFactory** *interface*
***Syntax*:**

```
Session ses= sef.openSession();
```

g. Create an implementation object of **Transaction** *interface* present in org.hibernate package by using a factory / helper method called *beginTransaction( )* which is declared inside **Session** *interface*
***Syntax*:**

```
Transaction tran= ses.beginTransaction();
```

h. Perform **CRUD** operation by using the reference of **Session**
***Syntax*:**

```
ses.save(Object);
```

`save(Object)` is declared in **Session** *interface* responsible to save / insert data's into the Database (only objects of java bean class)

i. Commit the **Transaction** in order **to save** the data into the Database server
Syntax:

```
tran.commit()
```

j. Close the Session

Since the Session is considered to be a costly resource, so we need to close the costly resource

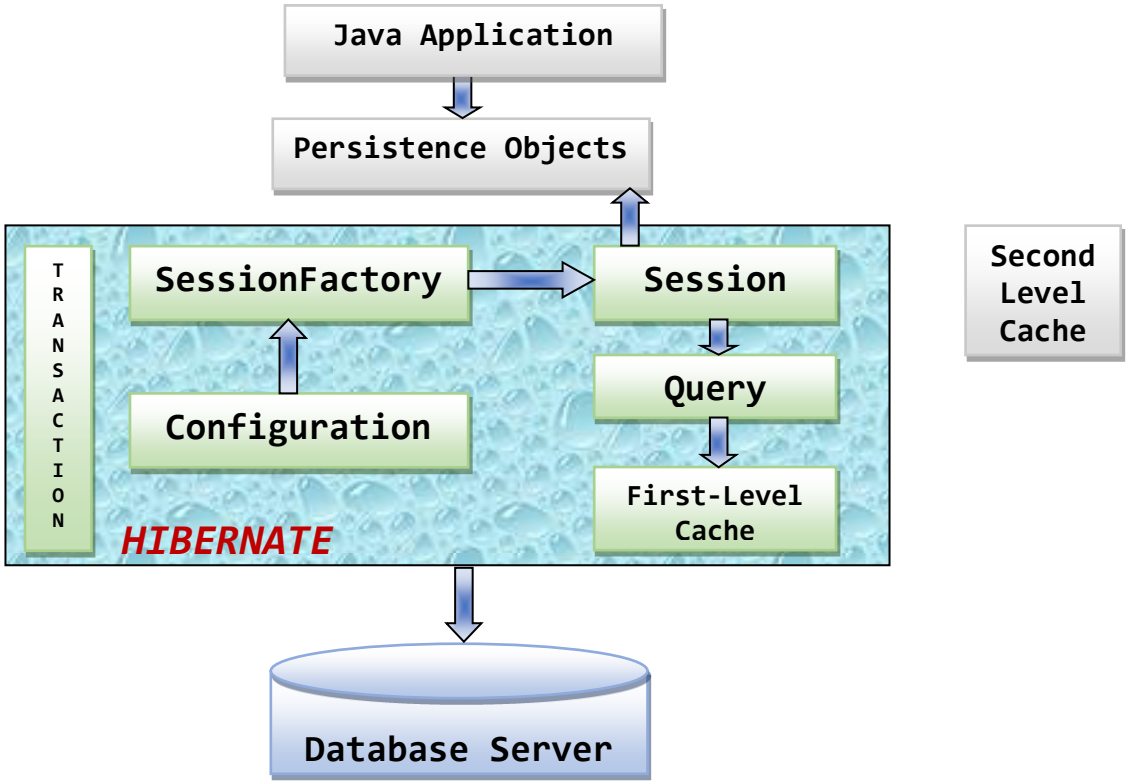
Syntax:

```
ses.close()
```

14. What is the role of configure( ) ?

`configure()` is responsible for loading and validating the configuration file called hibernate.cfg.xml

15. Explain hibernate architecture in brief?



i. **Configuration** is a *class* present in org.hibernate.cfg package which is used to load and validate the configuration file called hibernate.cfg.xml by calling a zero-parameterised **configure( )** method.

```
Configuration conf=new Configuration();
conf.configure();
```

ii. Whenever the configuration file has other name, then we use a parameterised configure( ).

```
conf.configure("Filename.xml");
```

iii. **SessionFactory** is an *interface* present in org.hibernate package for which an implementation object has to be created by using a factory / helper method called **buildSessionFactory( )** along with the reference of Configuration class.

iv. **SessionFactory** is used to establish a connection with the Database server.

v. For each Database server, there is only one **SessionFactory** present.

vi. **SessionFactory** always holds Second-Level Cache.

> Second-Level cache:- It always holds the constant data present in second-level configuration file (data present in SessionFactory Object)

vii. **Session** is an *interface* which is present in org.hibernate package for which an implementation object has to be created by calling a factory / helper method called **openSession( )** along with the reference of **SessionFactory** *interface*.

```
SessionFactory sef=confg.buildSessionFactory();
```

viii. **Session** *object* is light-weight object where 'n' number of Session object can be created for an application along with the same SessionFactory reference.

light-weight object ⇨ number of data present in it is minimum

ix. **Session** is always *Single-Threaded*.

x. **Session** object is responsible to carry the data from the Java Application to the Database server.

xi. **Session** object always holds First-Level cache.

> (**)Using **Session** object we perform **CRUD** operations

xii. **Transaction** is an interface present in org.hibernate package for which an implementation object has to created by calling a factory / helper method called beginTransaction( ) along with the reference of **Session** *interface*.

xiii. **Transaction** is used to achieve ***ACID properties*** / ***ACID rules***.

A	Atomicity
C	Consistency
I	Isolation
D	Durability

xiv. **commit( )** is used to save and reflect the changes in **tran.commit( )**

16. Explain cache mechanism in brief?

Cache memory: - it is temporary / buffer area which stores constant data or repeated data from the Database

> **Storage of cache memory:** cache memory is generally stored in **RAM**

✸ **Advantage of cache mechanism:-**

- Cache mechanism avoids the traffic between Java application and Database server. Hence, the performance of an application increases.

There are 2 different types of cache present namely;

i. First-Level Cache
ii. Second-Level Cache

i. First-Level Cache

- First-Level Cache is always associated with every Session Object which is enabled by default.
- First-Level Cache is the first cache that the hibernate consults before loading the Object from the Database.
- Once the Session is closed, all the data present in First-Level Cache are cleared.
- There are 2 different methods associated with First-Level Cache namely;
  1. evict( )
  2. clear( )

1. evict( ): it is used to remove a particular object from the cache associated with Session

2. clear( ): it is used to remove all the Objects from the cache associated with Session

ii. Second-Level Cache

- Second-Level Cache is always associated with SessionFactory Object which is not enabled by default, since it is an Optional cache.
- Since it is an optional cache, the developer has to set up the configuration
- There are different vendors who provides the implementation for second-level cache namely;
  a. EH cache
  b. OS cache
  c. JBOSS cache
  d. Swarm cache etc...

17. Difference between evict( ) and clear( ) ?

18.

evict( )	**clear( )**
It is used to remove a particular object from the cache associated with Session	It is used to remove all the objects from the cache associated with Session

Define the Generator and explain the types of Generator?

- "**Generator** is the one which is used to generate a primary key value based on Generation strategy / Generation Algorithm".
- It is a responsible for auto-generation of Primary Key.
- There are 2 different categories of generator present;
  i. JPA generator

ii. Hibernate generator

i. JPA Generator:- JPA generator supports 4 different types of Primary Key Generation Strategy which are as follows;

a. GenerationType.**AUTO**
b. GenerationType.**IDENTITY**
c. GenerationType.**SEQUENCE**
d. GenerationType.**TABLE**

a. GenerationType.**AUTO:**

- It is a ***default*** GenerationType which selects the generation strategy based on Database specific **dialect**
  [Based on the respective Database specifications, according to that primary key will be generated automatically]

Why default GenerationType**:-** For 2 important reasons;

1. Since it is supported by all the Database servers. Hence, the name ***default*** GenerationType
2. Whenever we **don't mention** the GenerationType, by ***default*** the GenerationType will be considered as GenerationType.**AUTO**

Syntax:

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

- Whenever we delete the data from actual table the record will not be deleted from comparison table
- Whenever we add new record on actual table the record will be added into comparison table

Comparison Table	Actual Table
1 ✓	1 ✓
2 ✓	2 ✓
3 ✓	3 ✗
4 ✓	4 ✓
	USER

b. GenerationType.**IDENTITY:**

- This GenerationType relies on an auto-incremented Database column

Syntax:

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

- It will check id present in Database and then will automatically increment the value based on that id in Database

c. GenerationType.**SEQUENCE**:

- This GenerationType is responsible for generating a primary key based on **sequence algorithm**
- This GenerationType is supported by only Oracle, IBM DB2, Postgres Database server

Syntax:

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

d. GenerationType.**TABLE**:

- This GenerationType is responsible for generating a primary key based on **Table algorithm**

Syntax:

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

```
1st value of id to be inserted id=1 for the 1st time
2nd time   : id= 32768
3rd time   : id= 65536
4th time   : id=98304
5th time   : id=131072
```

❖ Hibernate Generator:

- It supports many types of primary key Generation strategy which are as follows;
  a. Increment
  b. Foreign
  c. Identity
  d. Sequence
  e. Hilo
  f. Seqhilo
  g. Uuid
  h. Guid
  i. Assigned etc...

a. ***Increment***:-

- It generates the identifiers of type int, long or short that is unique where no other process is inserting the data into the same table
- Increment always auto-increments the value of Primary Key based on the maximum value of primary key present in the table
  [Maximum value of **Primary Key + 1**]

Syntax:

```
@Id
@GeneratedValue(generator="mygen")
```

```
@GenericGenerator(name="mygen",strategy= "increment")
```

b. ***Foreign***:-

It uses the identifiers of another associated object which is generally used in conjunction with **<OneToOne>** Primary Key Association

19. Define Association or Hibernate relationship?

"It represents the relationship between the objects of Java Bean Class / Entity Class / POJO Class"

Or

"It represents the relationship between 2 different tables"

✸ **Need for association:-** Association is needed **to store multiple-entities data** into a ***Single Database Table***

✸ **Problems:-**

1. ***Data Redundancy / Duplication of data***
2. ***Data Maintenance problem***

✸ **Types of Association:-** There are 4 different types of association present namely;

i. One To One
ii. One To Many
iii. Many To One
iv. Many To Many

20. What are the advantages of Association?

a. Data Redundancy problem is solved.
b. Navigation is possible with the help of Foreign Key.
c. Data Maintenance is easy