Hibernate

Test Questions and Assignments :

1. **What is Framework ?**
   - ▼ **Ans**
     - **It is body or platform where we have the pre-defined code which we can use in our program to reduce the code.**
     - **Framework is a hierarchy of class and Interfaces.**
     - **Framework is a body or a platform where we will get the solution for various technical problem.**
2. **Disadvantages of JDBC or Why we need to go Framework**
   - ▼ **Ans**
     1. **JDBC does not support automatic table creation.**
     2. **JDBC does not support generation of automatic Primary key.**
     3. **Duplicate code or boiler plate code.**
        - To save the record using JDBC we need to follow the below steps :-
          1. Load and Register the Driver.
          2. Establish the connection between Java application and Database Server.
          3. Create statement or Platform.
          4. Execute the Insert Query.
          5. Close all the costly resources.
     4. **To Update, Delete, Fetch we need to follow the same steps only the query is different. which with result in duplicate code or boiler plate code.**
     5. **JDBC does not support cache mechanism because of which traffic between java application and database server will increase and efficiency will decreases.**
3. **What is Cache memory ?**
   - ▼ **Ans**
     - **It is a temporary layer of storage which is used to store the data for the future purpose.**
4. **Types of Frameworks ?**
   - ▼ Ans

     We have 2 types of Frameworks

     - ▼ **Invasive Frameworks**
       - **If we are allowed to extends the classes or implement the Interfaces of a framework such type of framework is called as Invasive Framework.**

         Ex: EJB Framework
     - ▼ **Non- Invasive Framework**
       - **A framework which does not allows us to extends the classes or implements interface is called as Non-Invasive framework.**

         Ex: Hibernate and Spring

5. **What is POJO ? (Plain Old Java Object)**

   ▼ **Ans**

   - **It is a class which does not have any restrictions other then the restrictions which are imposed by Java.**

6. **What is the characteristics of POJO ?**

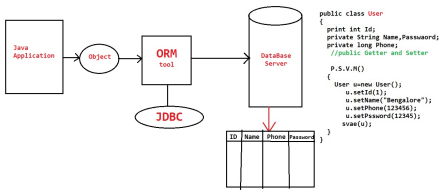
   ▼ **Ans**

   - **A POJA class should not implements any pre specified Interface.**
   - **A POJA class should not extends any pre specified classes.**
   - **A POJO class should not be Annotated with any pre specified annotation.**
   - **A POJA class must be public.**
   - **A POJA class should have a public no arguments constructor.**
   - **A POJO class should have public getters and setters for all the fields or variables.**

7. **What is ORM ? (Object Relational Mapping)**

   ▼ **Ans**

   - **The process of converting an object into Relational Model is called as Object Relational Mapping.**
     - **Relational Model is nothing but a row or record in Relational Database. (RDMS)**



8. **Specifications of ORM ?**

   ▼ **Ans**

   1. **Every Entity class or POJO class or bean class represents a Table in the Database.**
   2. **Every Variable of POJO class or bean class or Entity class represent a column in the Table.**
   3. **Every object of POJO class or bean class or Entity class represents a record in the Table.**

9. **What is Hibernate ?**

   ▼ **Ans**

   - **Hibernate is a Lightweight, Open source, Non-Invasive, ORM tool which is used for Object Relational Mapping.**

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

    ▼ **Ans**

    1. **It is a lightweight framework because of it is POJO implementation.**
    2. **It is open source framework.**
    3. **Using hibernate we can create the tables automatically.**
    4. **Hibernate provides the strategy for generation of primary key.**
    5. **Hibernate support cache mechanism. hibernate support 2 level of cache**
       a. **1st level cache**
       b. **2nd level cache**
    6. **Database Independent Queries**

- **Hibernate support HQL (Hibernate Query Language) because of which we can write Database Independent Queries with the help of Dialect.**
- **Dialect is used for the conversion of HQL queries into SQL Queries of the respective database server.**

11. **Requirements for Hibernate ?**

▼ **Ans**

1. **Eclipse IDE for Enterprise Java and Web Developments.**
2. **MySQL Database Server.**
3. **SQL Yog / MySQL Workbench**

12. **Rules to create an Entity class ?**

▼ **Ans**

1. **Entity class must be public and non-abstract.**
2. **There must be a public no argument constructor.**
3. **Every field should have public getters and setters.**
4. **Entity class should have a variable which represents Primary Key.**

13. **Steps to create Hibernate Project ?**

▼ **Ans**

1. Create simple maven project
   a. Open Eclips with a JEE perspective.
   b. Click on MAVEN project —> Click on the check box of (skip archetype selection)
2. Add dependencies in the pom.xml
   a. Type in Google “maven repository” —> select first link or click this link “https://mvnrepository.com/”
   b. Type in Search bar “Hibernate Core Relocation” —> Select Hibernate Core Relocation (most used)
   c. Select 5.6.5 file —> copy —> past it in pom.xml with dependencies tag.
   d. And again do same thing —> search “SQL Connector” —> select 8.0.28 —> copy —> past it in pom.xml in dependencies tag.
   e. Then save the project, It will download jar files. We can see that in our project files with name (Maven Dependencies)
3. Create an entity class to represent specific table.
   a. Create class with private variables in (src/main/java).
   b. Generate getter and setter for every variables by shortcut (Go to source —> click on generate getter and setter)
4. Create Hibernate configuration file (hibernate.cfg.xml)
   a. Create XML file in (src/main/java) and name of the file should be hibernate.cfg.xml
   b. For this we will get the code in “https://github.com/sathishnyadav”
   c. Go to that link and open supporting file and select hibernate.cfg.xml
   d. From there copy the entire code and past it on our hibernate.cfg.xml file.
   e. Remove the duplicate lines present in that code.
   f. Give the database name, username and password of database.
5. Create hibernate mapping file saved with extension (.hbm.xml)
   a. Create XML file in (src/main/java) and name of the file should be classname.hbm.xml

b. For this we will get the code in “https://github.com/sathishnyadav”

c. Go to that link and open supporting file and select Student.cfg.xml

d. Copy the code only from line 1 to line 4 —> past it in our XML file.

e. For hibernate mapping <hibernate-mapping> is the root tag.

f. Inside <hibernate-mapping>, we have to specify the class with FQCN and table.

g. We have to specify the primary key. using <id></id> tag

- We have to link configure file with mapping file —> open configuration file —> using <mapping resource=”classname.hbm.xml”> tag.

▼ pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.Jsp</groupId>
  <artifactId>Hibarnate-Demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
    <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
    <dependency>
      <groupId>org.hibernate</groupId>
      <artifactId>hibernate-core</artifactId>
      <version>5.6.5.Final</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.28</version>
    </dependency>
  </dependencies>
</project>
```

▼ User.java

```
package org.Jsp;

public class User
{
  private int id;
  private String name,password;
  private long phone;
  public int getId()
  {
    return id;
  }
  public void setId(int id)
  {
    this.id = id;
  }
  public String getName()
  {
    return name;
  }
  public void setName(String name)
  {
    this.name = name;
  }
  public String getPassword()
  {
    return password;
  }
  public void setPassword(String password)
  {
    this.password = password;
  }
  public long getPhone()
  {
```

```
    return phone;
  }
  public void setPhone(long phone)
  {
    this.phone = phone;
  }
}
```

▼ hibernate.cfg.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
  <session-factory>
    <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property name="connection.url">jdbc:mysql://localhost:3306/hibarnate_demo</property>
    <property name="dialect">org.hibernate.dialect.MySQL8Dialect</property>
    <property name="connection.username">root</property>
    <property name="connection.password">admin</property>

    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="hbm2ddl.auto">update</property>
    <mapping resource="User.hbm.xml" />
  </session-factory>
</hibernate-configuration>
```

▼ User.hbm.xml

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
  <class name="org.Jsp.User" table="User">
    <id name="id" column="id">
      <generator class="assigned"></generator>
    </id>
    <property name="name"></propert>
    <property name="phone"></property>
    <property name="password"></property>
  </class>
</hibernate-mapping>
```

14. **Steps to add the Dependencies ?**

▼ **Ans**

1. Browse maven repository in Browse.
2. Open “https://mvnrepository.com/”
3. search “Hibernate Core Relocation” and Open hibernate core relocation
4. Select the preferred version (5.6.5) and open
5. copy the dependency and past it in pom.xml between <dependencies></dependencies>
6. Follow the same to add MySQL-Connector-java with preferred version (8.0.28)

15. **org.hibernate.cfg.Configuration ?**

▼ Ans

- **It is a class belongs to Hibernate Framework used to configure the resources.**

Methods of Configuration class

1. configure()
    - This is a overloaded method. It is used to configure the resources and returns the reference of org.hibernate.cfg.Configuration.
2. buildSessionFactory()
    - It is a non-static method present in Configuration class. This method creates an implementation object of org.hibernate.SessionFactory and returns the reference. So the return type of this method is SessionFactory.

16. **org.hibernate.cfg.SessionFactory ?**

    ▼ Ans

    - It is an Interface. It is the Factory for org.hibernate.Session. This method creates an implementation object of org.hibernate.Session and returns the refereneces. So the return type of this method Session Interface.

    openSession()

    - It is a Factory or helper method present inside SessionFactory Interface.

17. **org.hibernate.Session ?**

    ▼ Ans

    - It is an Interface present inside hibernate package. It is used to manage the Entity with the help of built-In methods.

    Session Interface methods

    1. save(Object)
    2. update(Object)
    3. delete(Object)
    4. find(class<T> , PK)

18. **beginTracsection() method ?**

    ▼ Ans

    - It is a Factory or helper method present in Session Interface. This method creates an implementation object for org.hibernate.Tracsection and returns the reference. So the return type of this method is Transection Interface.

19. **org.hibernate.Transection ?**

    ▼ Ans

    - It is an Interface. It is used to control the Transections.

    1. commit()
        - It is method present in Transection Interface used to commit the Transections.

20. **Code-1 to Save the data into the Database using Session Interface ?**

    ▼ Ans

    ▼ pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.jsp</groupId>
  <artifactId>Hibernate-demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
    <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
    <dependency>
      <groupId>org.hibernate</groupId>
```

```
      <artifactId>hibernate-core</artifactId>
      <version>5.6.5.Final</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
    <dependency>
      <groupId>mysql</groupId>
      <artifactId>mysql-connector-java</artifactId>
      <version>8.0.28</version>
    </dependency>
  </dependencies>
</project>
```

▼ **User.java**

```
package org.jsp;

public class User
{
  private int Id;
  private String Name;
  private int Age;
  private String Color;

  public int getId() {
    return Id;
  }
  public void setId(int id) {
    Id = id;
  }
  public String getName() {
    return Name;
  }
  public void setName(String name) {
    Name = name;
  }
  public int getAge() {
    return Age;
  }
  public void setAge(int age) {
    Age = age;
  }
  public String getColor() {
    return Color;
  }
  public void setColor(String color) {
    Color = color;
  }
}
```

▼ **hibernate.cfg.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
  <session-factory>
    <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property name="connection.url">jdbc:mysql://localhost:3306/manu</property>
    <property name="dialect">org.hibernate.dialect.MySQL8Dialect</property>
    <property name="connection.username">root</property>
    <property name="connection.password">admin</property>

    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="hbm2ddl.auto">update</property>
    <mapping resource="User.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

▼ **User.hbm.xml**

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
   <class name="org.jsp.User" table="User">
   <id name="Id" column="Id">
   <generator class="assigned"></generator>
   </id>
   <property name="name"></property>
    <property name="age"></property>
     <property name="color"></property>
   </class>
</hibernate-mapping>
```

▼ **SaveUser.java**

```
package org.jsp;

import org.hibernate.*;
import org.hibernate.cfg.Configuration;

public class SaveUser
{
  public static void main(String[] args)
  {
    User u=new User();
    u.setId(1);
    u.setName("Manu");
    u.setAge(23);
    u.setColor("Black");

    Configuration c=new Configuration();
    c.configure();

    SessionFactory sf=c.buildSessionFactory();
    Session s=sf.openSession();
    Transaction t=s.beginTransaction();
    s.save(u);  //For update the record. Just use s.update(u);
    t.commit();
  }
}
```

21. **Possible Exceptions ?**

▼ **Ans**

1. **Unknow database Exception —> “check the database server”**
2. **could not locate cfg.xml —> “check the spelling of hibernate configuration or check the folder where it is created”**
3. **Unknown mapping resources —> “check the spelling hibernate mapping file or folder”**
4. **Duplicate entry for key primary —> “check the primary key”**
5. **parse error → “check xml files”**
6. **could not locate getter —> “check the variable name and properties name in hibernate mapping file”**

22. **Code-2 to Save the data into the Database using Session Interface ?**

▼ **Ans**

▼ **pom.xml**

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="h
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.Jsp</groupId>
  <artifactId>Hibernate-Demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
```

```
  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.6.5.Final</version>
</dependency>
  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.28</version>
</dependency>

  </dependencies>
</project>
```

▼ **Person.java**

```
package org.Jsp;

public class Person
{
  private int id;
    private String name;
    private int weight;
    private int height;
    private long phone;
    public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public int getWeight() {
    return weight;
  }
  public void setWeight(int weight) {
    this.weight = weight;
  }
  public int getHeight() {
    return height;
  }
  public void setHeight(int height) {
    this.height = height;
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
}
```

▼ **Person.hbm.xml**

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
   <class name="org.Jsp.Person" table="Person">
   <id name="id" column="id">
   <generator class="assigned"></generator>
   </id>
   <property name="name"></property>
    <property name="weight"></property>
     <property name="height"></property>
      <property name="phone"></property>
```

```
   </class>
</hibernate-mapping>
```

▼ **hibernate.cfg.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
  <session-factory>
    <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property name="connection.url">jdbc:mysql://localhost:3306/manu</property>
    <property name="dialect">org.hibernate.dialect.MySQL8Dialect</property>
    <property name="connection.username">root</property>
    <property name="connection.password">admin</property>

    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="hbm2ddl.auto">update</property>
    <mapping resource="Person.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

▼ **SavePerson.java**

```
package org.Jsp;

import org.hibernate.*;
import org.hibernate.cfg.Configuration;

public class SavePerson
{
  public static void main(String[] args)
  {
    Person p = new Person();
    p.setId(1);
    p.setName("ABCD");
    p.setHeight(12);
    p.setWeight(60);
    p.setPhone(12345);

    Configuration c=new Configuration();
    c.configure();

    SessionFactory sf=c.buildSessionFactory();
    Session s=sf.openSession();
    Transaction t=s.beginTransaction();
    s.save(p);
    t.commit();
  }
}
```

23. **find(class<T> , Object Primary Key) method ?**

▼ **Ans**

- **This method will return the reference of the object, if a record is present in the table with passed primary key otherwise, It will return null.**

24. **Code-3 to Fetch the data from the Database using find() method ?**

▼ **Ans**

▼ **pom.xml**

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="h
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.Jsp</groupId>
  <artifactId>Hibernate-Demo</artifactId>
```

```
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.6.5.Final</version>
</dependency>
  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.28</version>
</dependency>

  </dependencies>
</project>
```

▼ Person.java

```
package org.Jsp;

public class Person
{
  private int id;
    private String name;
    private int weight;
    private int height;
    private long phone;
    public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public int getWeight() {
    return weight;
  }
  public void setWeight(int weight) {
    this.weight = weight;
  }
  public int getHeight() {
    return height;
  }
  public void setHeight(int height) {
    this.height = height;
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
}
```

▼ Person.hbm.xml

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
   <class name="org.Jsp.Person" table="Person">
   <id name="id" column="id">
   <generator class="assigned"></generator>
   </id>
   <property name="name"></property>
    <property name="weight"></property>
     <property name="height"></property>
```

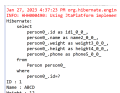
```
      <property name="phone"></property>
   </class>
</hibernate-mapping>
```

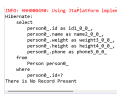
▼ **hibernate.cfg.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
  <session-factory>
    <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property name="connection.url">jdbc:mysql://localhost:3306/manu</property>
    <property name="dialect">org.hibernate.dialect.MySQL8Dialect</property>
    <property name="connection.username">root</property>
    <property name="connection.password">admin</property>

    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="hbm2ddl.auto">update</property>
    <mapping resource="Person.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

▼ **SavePerson.java**

```
package org.Jsp;

import org.hibernate.*;
import org.hibernate.cfg.Configuration;

public class SavePerson
{
  public static void main(String[] args)
  {
    Configuration c=new Configuration();
    c.configure();

    SessionFactory sf=c.buildSessionFactory();
    Session s=sf.openSession();

    Person p = s.find(Person.class, 2);
    if(p!=null)
    {
      System.out.println("ID : "+p.getId());
      System.out.println("Name : "+p.getName());
      System.out.println("Height : "+p.getHeight());
      System.out.println("Weight : "+p.getWeight());
      System.out.println("Phone Number : "+p.getPhone());
    }
    else
    {
      System.out.println("There is No Record Present");
    }
  }
}
```





25. **saveOrUpdate(Object) method ?**

▼ Ans

- **This method will update the record, If the Primary Key is already present. But, If in case Primary Key is not present, then it will create a new record based on Primary Key value.**

26. **Code-4 to Save the data or Update the data using saveOrUpdate() method ?**

▼ Ans

▼ pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="h
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.Jsp</groupId>
  <artifactId>Hibernate-Demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.6.5.Final</version>
</dependency>
  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.28</version>
</dependency>

  </dependencies>
</project>
```

▼ Person.java

```
package org.Jsp;

public class Person
{
  private int id;
    private String name;
    private int weight;
    private int height;
    private long phone;
    public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public int getWeight() {
    return weight;
  }
  public void setWeight(int weight) {
    this.weight = weight;
  }
  public int getHeight() {
    return height;
  }
  public void setHeight(int height) {
    this.height = height;
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
}
```

▼ **Person.hbm.xml**

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
   <class name="org.Jsp.Person" table="Person">
   <id name="id" column="id">
   <generator class="assigned"></generator>
   </id>
   <property name="name"></property>
    <property name="weight"></property>
     <property name="height"></property>
      <property name="phone"></property>
   </class>
</hibernate-mapping>
```

▼ **hibernate.cfg.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
  <session-factory>
    <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property name="connection.url">jdbc:mysql://localhost:3306/manu</property>
    <property name="dialect">org.hibernate.dialect.MySQL8Dialect</property>
    <property name="connection.username">root</property>
    <property name="connection.password">admin</property>

    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="hbm2ddl.auto">update</property>
    <mapping resource="Person.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

▼ **SavePerson.java**

```
package org.Jsp;

import org.hibernate.*;
import org.hibernate.cfg.Configuration;

public class SavePerson
{
  public static void main(String[] args)
  {
    Configuration c=new Configuration();
    c.configure();

    SessionFactory sf=c.buildSessionFactory();
    Session s=sf.openSession();
    Transaction t=s.beginTransaction();

    Person p = new Person();
    p.setId(1);
    p.setName("Good Vibes Only");
    p.setHeight(0);
    p.setWeight(0);
    p.setPhone(12345);

    s.saveOrUpdate(p);
    t.commit();
}
```

27. **Code-5 to Delete the data using delete() method ?**

▼ **Ans**

▼ **pom.xml**

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="h
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.Jsp</groupId>
  <artifactId>Hibernate-Demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.6.5.Final</version>
</dependency>
  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.28</version>
</dependency>

  </dependencies>
</project>
```

▼ **Person.java**

```
package org.Jsp;

public class Person
{
  private int id;
    private String name;
    private int weight;
    private int height;
    private long phone;
    public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public int getWeight() {
    return weight;
  }
  public void setWeight(int weight) {
    this.weight = weight;
  }
  public int getHeight() {
    return height;
  }
  public void setHeight(int height) {
    this.height = height;
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
}
```

▼ **Person.hbm.xml**

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
```

```
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
   <class name="org.Jsp.Person" table="Person">
   <id name="id" column="id">
   <generator class="assigned"></generator>
   </id>
   <property name="name"></property>
    <property name="weight"></property>
     <property name="height"></property>
      <property name="phone"></property>
   </class>
</hibernate-mapping>
```

▼ **hibernate.cfg.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
  <session-factory>
    <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property name="connection.url">jdbc:mysql://localhost:3306/manu</property>
    <property name="dialect">org.hibernate.dialect.MySQL8Dialect</property>
    <property name="connection.username">root</property>
    <property name="connection.password">admin</property>

    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="hbm2ddl.auto">update</property>
    <mapping resource="Person.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

▼ **DeletePersonRecord.java**

```
package org.Jsp;

import org.hibernate.*;
import org.hibernate.cfg.Configuration;

public class DeletePersonRecord
{
  public static void main(String[] args)
  {
    Configuration c=new Configuration();
    c.configure();

    SessionFactory sf=c.buildSessionFactory();
    Session s=sf.openSession();
    Transaction t=s.beginTransaction();

    Person p=s.find(Person.class, 1);
    if(p!=null)
    {
      s.delete(p);
      t.commit();
    System.out.println("Deleted the Record");
    }
    else
    {
      System.out.println("ID not found");
    }
  }
}
```

- **Telegram Group Link —> “tinyurl.com/SEHM20”**

28. **org.hibernate.createQuery() ?**

▼ **Ans**

- It is helper method present in Session Interface.
- This method will create the implementation object of org.hibernate.Query and returns the reference.
- So the return type of this method is Query.

29. **org.hibernate.Query ?**

▼ Ans

- It is an Interface. It is used to execute HQL (Hibernate Query Language)

30. **What is HQL ?**

▼ Ans

- It is a Query language which is similar to SQL.
- HQL queries are database independent.

SQL	HQL
1). select * from User	1). select u from User u
2). select * from user where id=?	2). select u from user u where u.id=?1

31. **getResultList() method ?**

▼ Ans

- It is a method present in Query Interface.
- This method will return the records for the particular query in form of java.util.List

32. **Code-6 to Fetch all the data Using HQL Queries ?**

▼ Ans

▼ pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="h
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.Jsp</groupId>
  <artifactId>Hibernate-Demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.6.5.Final</version>
</dependency>
  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.28</version>
</dependency>

  </dependencies>
</project>
```

▼ Person.java

```
package org.Jsp;

public class Person
{
  private int id;
    private String name;
    private int weight;
    private int height;
    private long phone;
```

```
    public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public int getWeight() {
    return weight;
  }
  public void setWeight(int weight) {
    this.weight = weight;
  }
  public int getHeight() {
    return height;
  }
  public void setHeight(int height) {
    this.height = height;
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
}
```

▼ **Person.hbm.xml**

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
   <class name="org.Jsp.Person" table="Person">
   <id name="id" column="id">
   <generator class="assigned"></generator>
   </id>
   <property name="name"></property>
    <property name="weight"></property>
     <property name="height"></property>
      <property name="phone"></property>
   </class>
</hibernate-mapping>
```

▼ **hibernate.cfg.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
  <session-factory>
    <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property name="connection.url">jdbc:mysql://localhost:3306/manu</property>
    <property name="dialect">org.hibernate.dialect.MySQL8Dialect</property>
    <property name="connection.username">root</property>
    <property name="connection.password">admin</property>

    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="hbm2ddl.auto">update</property>
    <mapping resource="Person.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

▼ **GetAllRecord.java**

```
package org.Jsp;

import java.util.List;

import org.hibernate.*;
import org.hibernate.cfg.Configuration;

@SuppressWarnings("all") //If we use this line, It will not show Yellow color lines
public class GetAllRecord
{
  public static void main(String[] args)
  {
    Session session = new Configuration().configure().buildSessionFactory().openSession();//This process is called as Method C
    String hql = "select p from Person p";
    Query<Person> q = session.createQuery(hql);
    List<Person> ps = q.getResultList();
    for(Person p : ps)
    {
      System.out.println("ID : "+p.getId());
      System.out.println("Name : "+p.getName());
      System.out.println("Heigth : "+p.getHeight());
      System.out.println("Weight : "+p.getWeight());
      System.out.println("Phone Number : "+p.getPhone());
      System.out.println("----------------------------------------");
    }
  }
}
```

33. **Code-7 to Verify the data Using HQL Queries ?**

▼ **Ans**

▼ **pom.xml**

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="h
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.Jsp</groupId>
  <artifactId>Hibernate-Demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.6.5.Final</version>
</dependency>
  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.28</version>
</dependency>

  </dependencies>
</project>
```

▼ **Person.java**

```
package org.Jsp;

public class Person
{
  private int id;
    private String name;
    private int weight;
    private int height;
    private long phone;
    public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
```

```
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public int getWeight() {
    return weight;
  }
  public void setWeight(int weight) {
    this.weight = weight;
  }
  public int getHeight() {
    return height;
  }
  public void setHeight(int height) {
    this.height = height;
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
}
```

▼ **Person.hbm.xml**

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
   <class name="org.Jsp.Person" table="Person">
   <id name="id" column="id">
   <generator class="assigned"></generator>
   </id>
   <property name="name"></property>
    <property name="weight"></property>
     <property name="height"></property>
      <property name="phone"></property>
   </class>
</hibernate-mapping>
```

▼ **hibernate.cfg.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
  <session-factory>
    <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property name="connection.url">jdbc:mysql://localhost:3306/manu</property>
    <property name="dialect">org.hibernate.dialect.MySQL8Dialect</property>
    <property name="connection.username">root</property>
    <property name="connection.password">admin</property>

    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="hbm2ddl.auto">update</property>
    <mapping resource="Person.hbm.xml"/>
  </session-factory>
</hibernate-configuration>
```

▼ **VarifyUser.java**

```
package org.Jsp;

import java.util.List;

import org.hibernate.*;
```

```
import org.hibernate.cfg.Configuration;

@SuppressWarnings("all")
public class VarifyUser
{
  public static void main(String[] args)
  {
    Session session = new Configuration().configure().buildSessionFactory().openSession();
    String qry = "select p from Person p where p.id=?1 and p.phone=?2";
    Query<Person> q = session.createQuery(qry);
    q.setParameter(1, 1);
    q.setParameter(2, 6l);
    List<Person> users = q.getResultList();
    if(users.size() > 0)
    {
      Person p = users.get(0);
      System.out.println("Login Sucessful");
      System.out.println("Name : "+p.getName());
      System.out.println("Weight : "+p.getWeight());
      System.out.println("Height : "+p.getHeight());
    }
    else
    {
      System.err.println("Invalid Phone Number and Id");
    }
  }
}
```

34. **Code-8 to Login Validation Using HQL Queries ?**

▼ Ans

▼ **pom.xml**

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="h
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.Jsp</groupId>
  <artifactId>Hibernate-Demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <dependencies>
  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.6.5.Final</version>
</dependency>
  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.28</version>
</dependency>

  </dependencies>
</project>
```

▼ **User.java**

```
package org.Jsp;

public class User
{
  private int Id;
  private String Name;
  private int Age;
  private String Color;
  private long Phone;
  private String Password;
  public int getId() {
    return Id;
  }
  public void setId(int id) {
    Id = id;
  }
  public String getName() {
```

```
    return Name;
  }
  public void setName(String name) {
    Name = name;
  }
  public int getAge() {
    return Age;
  }
  public void setAge(int age) {
    Age = age;
  }
  public String getColor() {
    return Color;
  }
  public void setColor(String color) {
    Color = color;
  }
  public long getPhone() {
    return Phone;
  }
  public void setPhone(long phone) {
    Phone = phone;
  }
  public String getPassword() {
    return Password;
  }
  public void setPassword(String password) {
    Password = password;
  }
}
```

▼ **User.hbm.xml**

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
"-//Hibernate/Hibernate Mapping DTD//EN"
"http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">
<hibernate-mapping>
   <class name="org.Jsp.User" table="User">
   <id name="Id" column="Id">
   <generator class="assigned"></generator>
   </id>
   <property name="name"></property>
    <property name="age"></property>
     <property name="color"></property>
     <property name="phone"></property>
     <property name="password"></property>
   </class>
</hibernate-mapping>
```

▼ **hibernate.cfg.xml**

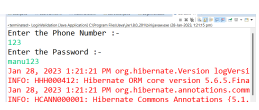
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-configuration PUBLIC
  "-//Hibernate/Hibernate Configuration DTD 3.0//EN"
  "http://www.hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>
  <session-factory>
    <property name="connection.driver_class">com.mysql.cj.jdbc.Driver</property>
    <property name="connection.url">jdbc:mysql://localhost:3306/manu</property>
    <property name="dialect">org.hibernate.dialect.MySQL8Dialect</property>
    <property name="connection.username">root</property>
    <property name="connection.password">admin</property>

    <property name="show_sql">true</property>
    <property name="format_sql">true</property>
    <property name="hbm2ddl.auto">update</property>
    <mapping resource="User.hbm.xml" />
  </session-factory>
</hibernate-configuration>
```

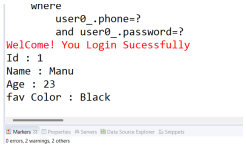
▼ **SaveUser.java (for Table creation & Insertion of Records)**

```
package org.Jsp;

import org.hibernate.*;
import org.hibernate.cfg.Configuration;

public class SaveUser
{
  public static void main(String[] args)
  {
    User u=new User();
    u.setId(1);
    u.setName("Manu");
    u.setAge(23);
    u.setColor("Black");
    u.setPhone(123);
    u.setPassword("manu123");


    Configuration c=new Configuration();
    c.configure();

    SessionFactory sf=c.buildSessionFactory();
    Session s=sf.openSession();
    Transaction t=s.beginTransaction();
    s.save(u);  //For update the record. Just use s.update(u);
    t.commit();
  }
}
```

▼ **LoginValidation.java (for Login Validation)**

```
package org.Jsp;

import java.util.*;
import org.hibernate.*;
import org.hibernate.cfg.Configuration;

@SuppressWarnings("all")
public class LoginValidation
{
  public static void main(String[] args)
  {
     Scanner sc = new Scanner(System.in);
     System.out.println("Enter the Phone Number :- ");
     long phonenum = sc.nextLong();
     System.out.println("Enter the Password :- ");
     String pass = sc.next();sc.close();

    Session session = new Configuration().configure().buildSessionFactory().openSession();
    String qry = "select u from User u where u.phone=?1 and u.password=?2";
    Query<User> q = session.createQuery(qry);
    q.setParameter(1, phonenum);
    q.setParameter(2, pass);
    List<User> users = q.getResultList();
    if(users.size() > 0)
    {
      User u = users.get(0);
      System.err.println("WelCome! You Login Sucessfully");
      System.out.println("Id : "+u.getId());
      System.out.println("Name : "+u.getName());
      System.out.println("Age : "+u.getAge());
      System.out.println("fav Color : "+u.getColor());
    }
    else
    {
      System.err.println("Invalid Phone Number or Id. Ok Prends Boii!!");
    }
  }
}
```





35. **What is JPA ?**

▼ **Ans**

- **JPA stands for Java Persistence API. It is specification which will provide a platform to directly deal with the Objects. As JPA is just a specification, It cannot do anything by it's self. So we need to make use of any one of the ORM tool, such as Hibernate or Ibatis. With the help of JPA, We can achieve Loose Coupling.**

36. **Steps to Create Project for Hibernate with JPA ?**

▼ **Ans**

1. **Create a Simple Maven Project.**
2. **Add the Dependencies.**
3. **Create a Entity class**
4. **Right click on scr/main/resources and create a folder with the name META-INF.**
5. **Right click on META-INF and create a new configuration file with the name persistence.xml**
   - "https://github.com/sathishnyadav" —> supporting-files —>persistence.xml —> copy entire code and past it in our persistence.xml —> remove line no.9 (not required)

▼ **persistence.xml**

```
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
  http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
  version="2.1">

  <persistence-unit name="dev">
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <properties>
      <property name="javax.persistence.jdbc.driver"
        value="com.mysql.cj.jdbc.Driver" />
      <property name="javax.persistence.jdbc.url"
        value="jdbc:mysql://localhost:3306/manu_jpa" />
      <property name="javax.persistence.jdbc.user" value="root" />
      <property name="javax.persistence.jdbc.password"
        value="admin" />
      <property name="hibernate.show_sql" value="true" />

      <property name="hibernate.hbm2ddl.auto" value="update" />
      <property name="hibernate.dialect"
        value="org.hibernate.dialect.MySQL8Dialect" />
    </properties>
  </persistence-unit>
</persistence>
```

▼ **TestConfiguration.java**

```
package org.jsp;

import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class TestConfiguration
{
  public static void main(String[] args)
  {
      EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
      System.out.println(factory);
```

```
  }
}
---------------OUTPUT--------------
org.hibernate.internal.SessionFactoryImpl@4372b9b6
```

- **SessionFactory interface ——extends—> EntityManagerFactory interface.**
- **SessionFactory is an Interface, We cannot create an object. So for that SessionFactory's implementation class will give an object. SessionFactory's implementation class is SessionFactoryImpl. Since SessionFactory extends EntityManagerFactory, SessionFactoryImpl helper class will also give an implementation object for EntityManagerFactory due to IS-A Relationship. This we can see it in above code.**

37. **javax.persistence.Persistenece ?**

▼ **Ans**

- **It is a helper class present in JPA used to create EntityManagerFactroy.**

38. **createEntityFactory(String persisteceUnit , name) ?**

▼ **Ans**

- **This is a factory or helper method present inside Persistence class. This creates an implementation object for EntityManagerFactory and return the reference. Return type of this method is EntityManagerFactory interface.**

39. **javax.persistence.EntityManagerFactory ?**

▼ **Ans**

- **It is an Interface present in JPA. It is the factory for EntityManager.**

40. **createEntityManager() ?**

▼ **Ans**

- **It is a factory method or helper method present inside EntityManagerFactory Interface. This method creates an implementation object for javax.persistence.EntityManager and returns the reference. Return type of this method is EntityManager.**

41. **javax.persistenece.EntityManager ?**

▼ **Ans**

- **It is an Interface present in JPA used to manage the Entity with the help built-in method.**

1. **getTransaction()**
   - **It is a factory or helper method present inside EntityManager Interface. used to create an implementation object of javax.persistenece.EntityTransaction returns the referenece. Return type of this method EntityTracsaction interface**
2. **persist(Object)**
   - **It is used to persist a record into persistence system(database). This method throws SQLSyntaxErrorException, if the primary key already present with a message duplicate entry for key primary.**
3. **merge(Object)**
   - **This method will merge the state of an object (update) with record in the table, if primary key is present.**
   - **This method will persist the object into the database, if primary key is not present.**
4. **find(class<T> required type , Object Primary Key)**
   - **This method returns the reference of the object, if any record found with the passed primary key otherwise returns null.**
   - **If we pass the required type has a class which is not an entity class then we will get unknow entity Exception.**

- Return type of the **find()** method is same as entity class.

5. **remove(Object Entity)**
   - This method will accept an entity and remove it from the table.
   - If we pass null for remove method we will get IlligelArgumentException with a message “attempt to create delete event with null entity”
   - If we pass the object which is not an entity we will get IlligelArgumentException with a message “unknow entity”

42. **javax.persistence.EntityTransaction ?**

▼ **Ans**

- It is an Interface present in JPA used to control the transaction after an Entity.

1. **begin()**
   - It is a non-static method present in Entity. It is used to begin the transaction.
2. **commit()**
   - It is used to commit the transaction.

43. **javax.persistence.Entity ?**

▼ **Ans**

- It is class level annotation present in JPA used to map a Entity class with database Table.

44. **javax.persistence.Id ?**

▼ **Ans**

- It is a field level annotation present in JPA used to mark the field as a Primary Key.

45. **Code-9 to Save the data Using JPA ?**

▼ **Ans**

▼ **persistence.xml**

```
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
  http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
  version="2.1">

  <persistence-unit name="dev">
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <properties>
      <property name="javax.persistence.jdbc.driver"
        value="com.mysql.cj.jdbc.Driver" />
      <property name="javax.persistence.jdbc.url"
        value="jdbc:mysql://localhost:3306/manu_jpa" />
      <property name="javax.persistence.jdbc.user" value="root" />
      <property name="javax.persistence.jdbc.password"
        value="admin" />
      <property name="hibernate.show_sql" value="true" />

      <property name="hibernate.hbm2ddl.auto" value="update" />
      <property name="hibernate.dialect"
        value="org.hibernate.dialect.MySQL8Dialect" />
    </properties>
  </persistence-unit>
</persistence>
```

▼ **Employee.java**

```
package org.jsp;
```

```
import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Employee
{
  @Id
  private int id;
  private String name,designation,password;
  private double salary;
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getDesignation() {
    return designation;
  }
  public void setDesignation(String designation) {
    this.designation = designation;
  }
  public String getPassword() {
    return password;
  }
  public void setPassword(String password) {
    this.password = password;
  }
  public double getSalary() {
    return salary;
  }
  public void setSalary(double salary) {
    this.salary = salary;
  }
}
```

▼ **SaveEmployee.java**

```
package org.jsp;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class SaveEmployee
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    EntityTransaction transaction = manager.getTransaction();
    Employee e1 = new Employee();
    e1.setId(1);
    e1.setName("Ragnar");
    e1.setDesignation("King");
    e1.setSalary(12345678);
    e1.setPassword("Ragnar@123");

    manager.persist(e1);
    transaction.begin();
    transaction.commit();
  }
}
```

46. **Code-10 to Update the data Using JPA ?**

▼ **Ans**

- **Same as Code-9 use merge() method.**

▼ **UpateEmployee**

```
package org.jsp;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class UpdateEmployee
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    EntityTransaction transaction = manager.getTransaction();
    Employee e1 = new Employee();
    e1.setId(1);
    e1.setName("Manu");
    e1.setDesignation("King");
    e1.setSalary(12345678);
    e1.setPassword("Ragnar@123");
    manager.merge(e1);
    transaction.begin();
    transaction.commit();
  }
}
```

47. **Code-11 to Fetch a single data Using JPA ?**

▼ **Ans**

▼ **FetchEmployee.java**

```
package org.jsp;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class FetchEmployee
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    Employee e = manager.find(Employee.class, 1);

    if(e!=null)
    {
      System.out.println("ID: "+e.getId());
      System.out.println("Name: "+e.getName());
      System.out.println("Designation: "+e.getDesignation());
      System.out.println("Salary: "+e.getSalary());
    }
    else
    {
      System.err.println("Invaild ID");
    }
  }
}
----------OUTPUT----------
ID: 1
Name: abd
Designation: King
Salary: 3000.0
```

48. **Code-12 to Delete the data Using JPA ?**

▼ Ans

▼ **DeleteEmployee.java**

```
package org.jsp;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class DeleteEmployee
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    EntityTransaction transaction = manager.getTransaction();
    Employee e1 = manager.find(Employee.class, 1);

    if(e1!=null)
    {
      manager.remove(e1);
      transaction.begin();
      transaction.commit();
    }
    else
    {
      System.err.println("Record is not Present");
    }
  }
}
```

49. **Code-13 to Fetch data by Designation Using JPA and JPQL Queries ?**

▼ **Ans**

▼ **FetchEmployeeByDesignation.java**

```
package org.jsp;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;

public class FetchEmployeeByDesignation
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    String qry = "select e from Employee e where e.designation=?1";
    Query q = manager.createQuery(qry);
    q.setParameter(1, "softwar");
    List<Employee> es = q.getResultList();

    if(es.size()!=0)
    {
      for(Employee e : es)
      {
        System.out.println("ID : "+e.getId());
        System.out.println("NAME : "+e.getName());
        System.out.println("Designation : "+e.getDesignation());
        System.out.println("SALARY : "+e.getSalary());
        System.out.println("------------------------------");
      }
    }
    else
    {
      System.out.println("Invalid Desination");
    }

  }
}
```

50. **Code-14 to Fetch data by Name Using JPA and JPQL Queries ?**

▼ Ans

▼ FetchEmployeeByName.java

```
package org.jsp;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;

public class FetchEmployeeByName
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    String qry = "select e from Employee e where e.name=?1";
    Query q = manager.createQuery(qry);
    q.setParameter(1, "Ragnar");
    List<Employee> es = q.getResultList();

    if(es.size()!=0)
    {
      for(Employee e : es)
      {
        System.out.println("ID : "+e.getId());
        System.out.println("NAME : "+e.getName());
        System.out.println("Designation : "+e.getDesignation());
        System.out.println("SALARY : "+e.getSalary());
        System.out.println("-------------------------------");
      }
    }
    else
    {
      System.out.println("Invaild Name! Please Provide Correct Name");
    }


  }
}
```

51. **Code-15 to Fetch data by Salary Using JPA and JPQL Queries ?**

▼ Ans

▼ FetchEmployeeBySalary.java

```
package org.jsp;

import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;

public class FetchEmployeeBySalary
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    String qry = "select e from Employee e where e.salary=?1";
    Query q = manager.createQuery(qry);
    q.setParameter(1, 5465.0);
    List<Employee> es = q.getResultList();

    if(es.size()!=0)
    {
      for(Employee e : es)
      {
        System.out.println("ID : "+e.getId());
        System.out.println("NAME : "+e.getName());
        System.out.println("Designation : "+e.getDesignation());
```

```
        System.out.println("SALARY : "+e.getSalary());
        System.out.println("------------------------------");
      }
    }
    else
    {
     System.out.println("Invaild Salary! Please Provide Correct Salary");
    }
  }
}
```

52. **Code-16 to Fetch data by Id and Password Using JPA and JPQL Queries ?**

▼ **Ans**

▼ **FetchEmployeeByIdandPassword.java**

```
package org.jsp;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;

public class FetchEmployeeByIdandPassword
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    String qry = "select e from Employee e where e.id=?1 and e.password=?2";
    Query q = manager.createQuery(qry);
    q.setParameter(1, 5);
    q.setParameter(2, "Ragnar@123");
    List<Employee> es = q.getResultList();

    if(es.size()!=0)
    {
      for(Employee e : es)
      {
        System.out.println("ID : "+e.getId());
        System.out.println("NAME : "+e.getName());
        System.out.println("Designation : "+e.getDesignation());
        System.out.println("SALARY : "+e.getSalary());
        System.out.println("------------------------------");
      }
    }
    else
    {
      System.out.println("Invalid! Please Provide Correct ID and PAssword");
    }

  }
}
```

53. **javax.persistence.GeneratedValue ?**

▼ **Ans**

- **It is a field level annotation present in JPA. It is used for the generation of Primary Key. We can specify the strategy for the generation of Primary Key using javax.persistence.GenerationType.**

54. **javax.persistence.GeneratedType ?**

▼ **Ans**

- **It is an Enum (Enumeration) present in JPA. it will provide the strategy for the generation of Primary Key, following are different strategies for available**

1. **GenerationType.IDENTITY**
2. **GenerationType.SQUENCE**

3. **GenerationType.AUTO**
4. **GenerationType.TABLE**

55. **Code-17 to Save the data for Automatic Generation of Primary Key Using GeneratedValue and GeneratedType ?**

▼ **Ans**

▼ **persistence.xml**

```
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
  http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
  version="2.1">

  <persistence-unit name="dev">
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <properties>
      <property name="javax.persistence.jdbc.driver"
        value="com.mysql.cj.jdbc.Driver" />
      <property name="javax.persistence.jdbc.url"
        value="jdbc:mysql://localhost:3306/manu_jpa" />
      <property name="javax.persistence.jdbc.user" value="root" />
      <property name="javax.persistence.jdbc.password"
        value="admin" />
      <property name="hibernate.show_sql" value="true" />

      <property name="hibernate.hbm2ddl.auto" value="update" />
      <property name="hibernate.dialect"
        value="org.hibernate.dialect.MySQL8Dialect" />
    </properties>
  </persistence-unit>
</persistence>
```

▼ **Person.java**

```
package org.jsp;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;

@Entity
@Table (name="Person_data")
public class Person
{
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  @Column(nullable = false)
  private String name;
  @Column(nullable = false)
  private int age;
  @Column(nullable = false,unique = true)
  private long phone;
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public int getAge() {
    return age;
  }
  public void setAge(int age) {
    this.age = age;
```

```
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
}
```

▼ **SavePerson.java**

```
package org.jsp;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class SavePerson
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    EntityTransaction transaction = manager.getTransaction();
    Person p =new Person();
    p.setName("Ragnar");
    p.setAge(21);
    p.setPhone(7863);

    manager.persist(p);
    transaction.begin();
    transaction.commit();
    System.out.println("ID : "+p.getId());
  }
}
```

56. **Code-18 to Update the data for Automatic Generation of Primary Key Using GeneratedValue and GeneratedType ?**

▼ **Ans**

▼ **SavePerson.java**

```
package org.jsp;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class SavePerson
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    EntityTransaction transaction = manager.getTransaction();
    Person p =new Person();
    p.setName("Ragnar km");
    p.setAge(21);
    p.setPhone(7863);

    manager.merge(p);
    transaction.begin();
    transaction.commit();
    System.out.println("ID : "+p.getId());
  }
}
```

57. **Code-19 to Fetch the data for Automatic Generation of Primary Key Using GeneratedValue and GeneratedType ?**

▼ **Ans**

▼ **Fetch.java**

```
public class Fetch
{
  public static void main(String[] args)
  {
    EntityManager entityManager = Persistence.createEntityManagerFactory("dev").createEntityManager();
    EntityTransaction entityTransaction = entityManager.getTransaction();

    Person p = entityManager.find(Person.class, 1);
    if (p != null)
    {
      System.out.println(p.getId());
      System.out.println(p.getName());
      System.out.println(p.getPhone());
      System.out.println(p.getAge());
    }
    else
    {
      System.err.println("Invalid ID");
    }
  }
}
```

58. **Code-20 to Delete the data for Automatic Generation of Primary Key Using GeneratedValue and GeneratedType ?**

▼ **Ans**

▼ **DeletePerson.java**

```
public class Fetch
{
  public static void main(String[] args)
  {
    EntityManager entityManager = Persistence.createEntityManagerFactory("dev").createEntityManager();
    EntityTransaction entityTransaction = entityManager.getTransaction();

    Person p = entityManager.find(Person.class, 1);
    if (p != null)
    {
      entityManager.remove(p);
      entityTransaction.begin();
      entityTransaction.commit();
    }
    else
    {
      System.err.println("Invalid ID");
    }
  }
}
```

59. **Annotations ?**

▼ **Ans**

1. **@Entity**
2. **@ID**
3. **@Table**
4. **@Column**
5. **@Genereted**
6. **@GeneratedValue(strategy = GenerationType.)**

---

62. **What is Relationship ?**

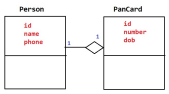
▼ **Ans**

- **The Association or connection between 2 objects is called as a Relationship. We have 2 types of Relationship.**

▼ HAS-A Relationship

- One object having the reference of another object is nothing but HAS-A Relationship or One object depending upon another object is called as HAS-A Relationship. We have many types of HAS-A Relationship

▼ One to One [1-1]



▼ Person.java

```
package org.Jspider.NormalApp;

public class Person
{
  private int id;
  private String name;
  private long phone;
  private PanCard pan;
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
  public PanCard getPan() {
    return pan;
  }
  public void setPan(PanCard pan) {
    this.pan = pan;
  }
}
```

▼ PanCard.java

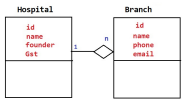
```
package org.Jspider.NormalApp;

import java.time.LocalDate;

public class PanCard
{
  private int id;
  private String number,state;
  private LocalDate dob;
  public int getId()
  {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getNumber() {
    return number;
  }
  public void setNumber(String number) {
    this.number = number;
  }
```

```
  public String getState() {
    return state;
  }
  public void setState(String state) {
    this.state = state;
  }
  public LocalDate getDob() {
    return dob;
  }
  public void setDob(LocalDate dob) {
    this.dob = dob;
  }
}
```

▼ SavePerson.java

```
package org.Jspider.NormalApp;

import java.time.LocalDate;

public class SavePerson
{
  public static void main(String[] args)
  {
    Person p = new Person();
    p.setId(1);
    p.setName("Heisenberg");
    p.setPhone(123);

    PanCard card = new PanCard();
    card.setId(1);
    card.setNumber("AGH13534AD");
    card.setState("Harapanahalli");
    card.setDob(LocalDate.parse("1992-07-11"));
  }
}
```

▼ One To Many or Many To One [1-n]



▼ Hospital.java

```
package org.Jspider.NormalApp;

import java.util.List;

public class Hospital
{
private int id;
private String name,founder,Gst;
private List<Branch> branch;
public int getId() {
  return id;
}
public void setId(int id) {
  this.id = id;
}
public String getName() {
  return name;
}
public void setName(String name) {
  this.name = name;
}
public String getFounder() {
  return founder;
}
public void setFounder(String founder) {
  this.founder = founder;
}
```

```
public String getGst() {
  return Gst;
}
public void setGst(String gst) {
  Gst = gst;
}
public List<Branch> getBranch() {
  return branch;
}
public void setBranch(List<Branch> branch) {
  this.branch = branch;
}
}
```

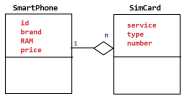
▼ **Branch.java**

```
package org.Jspider.NormalApp;

public class Branch
{
  private int id;
  private String name,email;
  private long phone;
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
  public String getEmail() {
    return email;
  }
  public void setEmail(String email) {
    this.email = email;
  }
}
```

▼ **SaveHospital.java**

```
package org.Jspider.NormalApp;

import java.util.ArrayList;
import java.util.List;

public class SaveHospital
{
  public static void main(String[] args)
  {
    Hospital hospital = new Hospital();
    hospital.setId(1);
    hospital.setName("Jessi Pinkman");
    hospital.setGst("AFS545FW");
    hospital.setFounder("ABC");

    Branch b1 =new Branch();
    b1.setId(1);
    b1.setEmail("abc99@gmail.com");
    b1.setName("Apollo");
    b1.setPhone(123);

    Branch b2 =new Branch();
    b1.setId(2);
    b1.setEmail("xyz99@gmail.com");
```

```
    b1.setName("Manipal");
    b1.setPhone(321);

    List<Branch> branchs = new ArrayList<Branch>();
    branchs.add(b1);
    branchs.add(b2);
    hospital.setBranch(branchs);
  }
}
```



▼ **SmartPhone.java**

```
package org.Jspider.NormalApp;

import java.util.List;

public class SmartPhone
{
private int id;
private String brand;
private int RAM;
private int price;
private List<SimCard> sim;
public int getId() {
  return id;
}
public void setId(int id) {
  this.id = id;
}
public String getBrand() {
  return brand;
}
public void setBrand(String brand) {
  this.brand = brand;
}
public int getRAM() {
  return RAM;
}
public void setRAM(int rAM) {
  RAM = rAM;
}
public int getPrice() {
  return price;
}
public void setPrice(int price) {
  this.price = price;
}
public List<SimCard> getSim() {
  return sim;
}
public void setSim(List<SimCard> sim) {
  this.sim = sim;
}
}
```

▼ **SimCard.java**

```
package org.Jspider.NormalApp;

public class SimCard
{
  private int id;
  private String service;
  private String type;
  private int number;
  public int getId() {
    return id;
  }
```

```
  public void setId(int id) {
    this.id = id;
  }
  public String getService() {
    return service;
  }
  public void setService(String service) {
    this.service = service;
  }
  public String getType() {
    return type;
  }
  public void setType(String type) {
    this.type = type;
  }
  public int getNumber() {
    return number;
  }
  public void setNumber(int number) {
    this.number = number;
  }

}
```

▼ **SaveSmartPhone.java**

```
package org.Jspider.NormalApp;

import java.util.ArrayList;
import java.util.List;

public class SaveSmartPhone
{
  public static void main(String[] args)
  {
    SmartPhone phone =new SmartPhone();
    phone.setId(1);
    phone.setBrand("Nokia 0080");
    phone.setRAM(512);
    phone.setPrice(800);


    SimCard card1 = new SimCard();
    card1.setId(123);
    card1.setService("Relience");
    card1.setType("Pre-Paid");
    card1.setNumber(12345678);

    SimCard card2 = new SimCard();
    card2.setId(321);
    card2.setService("BSNL");
    card2.setType("Post-Paid");
    card2.setNumber(87654321);

    List<SimCard> cards =new ArrayList<SimCard>();
    cards.add(card1);
    cards.add(card2);

    phone.setSim(cards);
  }
}
```

▼ **Many to Many**

▼ **IS-A Relationship**

63. **What is Association Mapping ?**

▼ **Ans**

- **The Process Establishing the relationship between 2 database tables is called as Association Mapping. We have 2 types Association Mapping.**
  - **Uni-Directional Mapping**

- Bi-Directional Mapping

- We can further classify Association Mapping into 4 types.

1. One to One
2. One to Many
3. Many to Many

64. @OneToOne UniDirectional ?

▼ Ans

- It is a field level annotation in JPA present inside javax.persistence package used to map one instance of a class with one instance of another.

65. Code-21 to Save the data Using @OneToOne Annotation ?

▼ Ans

- For automatic Database creation —use—> “DatabaseName?createDatabaseIfNotExist=true”

▼ persistence.xml

```
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
  http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
  version="2.1">

  <persistence-unit name="dev">
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <properties>
      <property name="javax.persistence.jdbc.driver"
        value="com.mysql.cj.jdbc.Driver" />
      <property name="javax.persistence.jdbc.url"
        value="jdbc:mysql://localhost:3306/one_to_one_uni?createDatabaseIfNotExist=true"/>
      <property name="javax.persistence.jdbc.user" value="root" />
      <property name="javax.persistence.jdbc.password"
        value="admin" />
      <property name="hibernate.show_sql" value="true" />

      <property name="hibernate.hbm2ddl.auto" value="update" />
      <property name="hibernate.dialect"
        value="org.hibernate.dialect.MySQL8Dialect" />
    </properties>
  </persistence-unit>
</persistence>
```

▼ Person.java

```
package org.jsp;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;import org.hibernate.resource.beans.internal.FallbackBeanInstanceProducer;

@Entity
@Table (name = "person")
public class Person
{
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  @Column (nullable = false)
  private String name,place;
  @Column (nullable = false, unique = true)
  private long phone;
  @OneToOne
  private PanCard pan;
```

```
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getPlace() {
    return place;
  }
  public void setPlace(String place) {
    this.place = place;
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
  public PanCard getPan() {
    return pan;
  }
  public void setPan(PanCard pan) {
    this.pan = pan;
  }
}
```

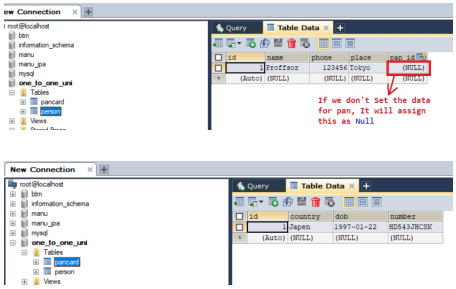
▼ **PanCard.java**

```
package org.jsp;

import java.time.LocalDate;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
@Entity
@Table (name = "PanCard")
public class PanCard
{
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  @Column(nullable = false,unique = true)
  private String number;
  @Column(nullable = false)
  private LocalDate dob;
  @Column(nullable = false)
  private String country;
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getNumber() {
    return number;
  }
  public void setNumber(String number) {
    this.number = number;
  }
  public LocalDate getDob() {
    return dob;
  }
  public void setDob(LocalDate dob) {
    this.dob = dob;
  }
  public String getCountry() {
    return country;
  }
```

```
  public void setCountry(String country) {
    this.country = country;
  }
}
```

▼ **SavePerson.java**

```
package org.jsp;

import java.time.LocalDate;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class SavePerson
{
  public static void main(String[] args)
  {
    Person p =new Person();
    p.setName("Proffsor");
    p.setPlace("Tokyo");
    p.setPhone(123456);

    PanCard card =new PanCard();
    card.setDob(LocalDate.parse("1997-01-22"));
    card.setNumber("HD543JHCSK");
    card.setCountry("Japen");

    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    EntityTransaction transaction = manager.getTransaction();

    manager.persist(p);
    manager.persist(card);
    transaction.begin();
    transaction.commit();
  }
}
```

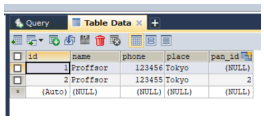


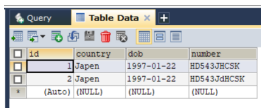
- Above code, We didn't Set the data for pan that is why it will assign Null. So now, We are going to Set the data for pan. It will store exact value in Database.

▼ **SavePerson.java**

```
package org.jsp;

import java.time.LocalDate;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class SavePerson
{
  public static void main(String[] args)
```

```
  {
    Person p =new Person();
    p.setName("Proffsor");
    p.setPlace("Tokyo");
    p.setPhone(123455);

    PanCard card =new PanCard();
    card.setDob(LocalDate.parse("1997-01-22"));
    card.setNumber("HD543JdHCSK");
    card.setCountry("Japen");
    p.setPan(card);///We need to set this by then only it will develop the RelationShip.

    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    EntityTransaction transaction = manager.getTransaction();

    manager.persist(p);
    manager.persist(card);
    transaction.begin();
    transaction.commit();
  }
}
```



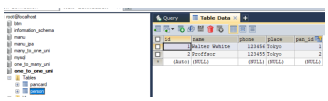


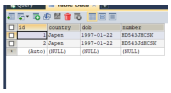
66. **Code-22 to Update the data of Person Using JPA and @OneToOne Annotation ?**

▼ **Ans**

▼ **UpdatePerson.java**

```
package org.jsp;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class UpdatePerson
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    EntityTransaction transaction = manager.getTransaction();

    PanCard card = manager.find(PanCard.class, 1);
    Person person = manager.find(Person.class, 1);

    person.setName("Walter Wwhite"); //Optional
    person.setPan(card);

    manager.merge(person);
    transaction.begin();
    transaction.commit();
  }
}
```





67. **Code-23 to Fetch the data of Person Using JPA and @OneToOne Annotation ?**

    ▼ **Ans**

68. **Code-24 to Delete the data of Person Using JPA and @OneToOne Annotation ?**

    ▼ **Ans**

69. **@OneToMany UniDirectional ?**

    ▼ **Ans**

    - **It is a field level annotation present in JPA.**
    - **It is present in javax.persistence package.**
    - **It is used to map a one instance of a class with many instances of another class.**

70. **Code-25 to Save the data Using @OneToMany Annotation ?**

    ▼ **Ans**

    ▼ **persistence.xml**

```
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
  http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
  version="2.1">

  <persistence-unit name="dev">
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <properties>
      <property name="javax.persistence.jdbc.driver"
        value="com.mysql.cj.jdbc.Driver" />
      <property name="javax.persistence.jdbc.url"
        value="jdbc:mysql://localhost:3306/one_to_many_uni?createDatabaseIfNotExist=true"/>
      <property name="javax.persistence.jdbc.user" value="root" />
      <property name="javax.persistence.jdbc.password"
        value="admin" />
      <property name="hibernate.show_sql" value="true" />

      <property name="hibernate.hbm2ddl.auto" value="update" />
      <property name="hibernate.dialect"
        value="org.hibernate.dialect.MySQL8Dialect" />
    </properties>
  </persistence-unit>
</persistence>
```

    ▼ **Hospital.java**

```
package org.jsp;

import java.util.List;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
@Entity
public class Hospital {
  @Id
  @GeneratedValue (strategy = GenerationType.IDENTITY)
  private int id;
  private String name, Gst, founder;
```

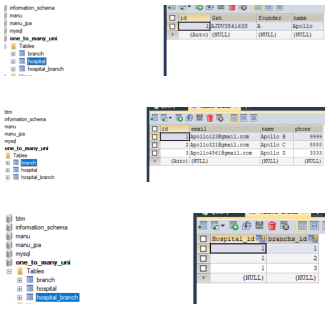
```
  @OneToMany
  private List<Branch> branchs;
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getGst() {
    return Gst;
  }
  public void setGst(String gst) {
    Gst = gst;
  }
  public String getFounder() {
    return founder;
  }
  public void setFounder(String founder) {
    this.founder = founder;
  }
  public List<Branch> getBranchs() {
    return branchs;
  }
  public void setBranchs(List<Branch> branchs) {
    this.branchs = branchs;
  }

}
```

▼ **Branch.java**

```
package org.jsp;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Branch {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  private String name, email;
  private long phone;
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getEmail() {
    return email;
  }
  public void setEmail(String email) {
    this.email = email;
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
}
```

▼ **SaveHospital.java**

```
package org.jsp;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class SaveHospital
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    EntityTransaction transaction = manager.getTransaction();
    Hospital hospital = new Hospital();
    hospital.setName("Apollo");
    hospital.setFounder("A");
    hospital.setGst("AJDV354163S");

    Branch b1 =new Branch();
    b1.setName("Apollo B");
    b1.setPhone(9999);
    b1.setEmail("Apollo123@gmail.com");

    Branch b2 =new Branch();
    b2.setName("Apollo C");
    b2.setPhone(8888);
    b2.setEmail("Apollo321@gmail.com");

    Branch b3 =new Branch();
    b3.setName("Apollo D");
    b3.setPhone(3333);
    b3.setEmail("Apollo4561@gmail.com");

    List<Branch> branchs = new ArrayList<Branch>(Arrays.asList(b1,b2,b3));
    hospital.setBranchs(branchs);
    manager.persist(hospital);
    manager.persist(b1);
    manager.persist(b2);
    manager.persist(b3);
    transaction.begin();
    transaction.commit();
  }
}
```



71. **Code-26 to Update the data in @OneToMany Annotation ?**

    ▼ **Ans**

72. **Code-27 to Fetch the data in @OneToMany Annotation ?**

    ▼ Ans

73. **Code-28 to Delete the data in @OneToMany Annotation ?**

    ▼ **Ans**

74. **@ManyToOne UniDirectional ?**

    ▼ **Ans**

    - **It is a field level annotation present in JPA.**
    - **It is present in javax.persistence package.**
    - **It is used to map a many instance of a class with one instances of another class.**

75. **Code-29 to Save the data Using @ManyToOne Annotation ?**

    ▼ **Ans**

    ▼ **persistence.xml**

    - **change only Database name —> "Many_To_One_Uni?createDatabaseIfNotExist=true"**

    ▼ **Hospital.java**

```
package org.jsp;

import java.util.List;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;

@Entity
public class Hospital {
  @Id
  @GeneratedValue (strategy = GenerationType.IDENTITY)
  private int id;
  private String name, Gst, founder;

  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getGst() {
    return Gst;
  }
  public void setGst(String gst) {
    Gst = gst;
  }
  public String getFounder() {
    return founder;
  }
  public void setFounder(String founder) {
    this.founder = founder;
  }


}
```

    ▼ **Branch.java**

```
package org.jsp;

import java.util.List;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

@Entity
public class Branch {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  private String name, email;
  private long phone;
  @ManyToOne
  private Hospital hos;
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getEmail() {
    return email;
  }
  public void setEmail(String email) {
    this.email = email;
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
  public Hospital getHos() {
    return hos;
  }
  public void setHos(Hospital hos) {
    this.hos = hos;
  }


}
```

▼ SaveHospital.java

```
package org.jsp;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class SaveHospital
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    EntityTransaction transaction = manager.getTransaction();
    Hospital hospital = new Hospital();
    hospital.setName("Apollo");
    hospital.setFounder("A");
    hospital.setGst("AJDV354163S");

    Branch b1 =new Branch();
    b1.setName("Apollo B");
```

```
    b1.setPhone(9999);
    b1.setEmail("Apollo123@gmail.com");
    b1.setHos(hospital);

    Branch b2 =new Branch();
    b2.setName("Apollo C");
    b2.setPhone(8888);
    b2.setEmail("Apollo321@gmail.com");
    b2.setHos(hospital);

    Branch b3 =new Branch();
    b3.setName("Apollo D");
    b3.setPhone(3333);
    b3.setEmail("Apollo4561@gmail.com");
    b3.setHos(hospital);

    manager.persist(hospital);
    manager.persist(b1);
    manager.persist(b2);
    manager.persist(b3);
    transaction.begin();
    transaction.commit();
  }
}
```

76. **Code-30 to Update the data in @ManyToOne Annotation ?**

    ▼ **Ans**

77. **Code-31 to Fetch the data in @ManyToOne Annotation ?**

    ▼ **Ans**

78. **Code-32 to Delete the data in @ManyToOne Annotation ?**

    ▼ **Ans**

79. **@ManyToMany UniDirectional ?**

    ▼ **Ans**

    - **It is field level annotation present in JPA.**
    - **It is present in javax.persistenece package.**
    - **It is used to map many instances of class with many instances of another class.**

80. **Code-33 to Save the data Using @ManyToMany Annotation ?**

    ▼ **Ans**

    ▼ **persistence.xml**

    - **change only Database name —> "Many_To_Many_Uni?createDatabaseIfNotExist=true"**

    ▼ **Student.java**

```
package org.jsp;

import java.util.List;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;

@Entity
public class Student {
  @Id
```

```
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  private String name;
  private long phone;
  private double perc;
  @ManyToMany
  private List<Course> courses;
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
  public double getPerc() {
    return perc;
  }
  public void setPerc(double perc) {
    this.perc = perc;
  }
  public List<Course> getCourses() {
    return courses;
  }
  public void setCourses(List<Course> courses) {
    this.courses = courses;
  }

}
```

▼ **Course.java**

```
package org.jsp;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Course
{
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  private String subject;
  private int duration;

  public int getId() {
    return id;
  }

  public void setId(int id) {
    this.id = id;
  }

  public String getSubject() {
    return subject;
  }

  public void setSubject(String subject) {
    this.subject = subject;
  }

  public int getDuration() {
    return duration;
  }
```

```
  public void setDuration(int duration) {
    this.duration = duration;
  }
}
```

▼ **SaveStudent.java**

```
package org.jsp;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import javax.transaction.Transaction;

public class SaveStudent
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    EntityTransaction transaction = manager.getTransaction();

    Student student1 = new Student();
    student1.setName("A");
    student1.setPhone(1233);
    student1.setPerc(63.36);

    Student student2 = new Student();
    student2.setName("B");
    student2.setPhone(12354333);
    student2.setPerc(61.36);

    Course c1 = new Course();
    c1.setSubject("Java");
    c1.setDuration(60);

    Course c2 = new Course();
    c2.setSubject("J2EE");
    c2.setDuration(40);

    List<Course> fors1 = new ArrayList<Course>();
    fors1.add(c1);
    fors1.add(c2);
    List<Course>  fors2 =new ArrayList<Course>(Arrays.asList(c1, c2));

    student1.setCourses(fors1);
    student2.setCourses(fors2);
    manager.persist(student1);
    manager.persist(student2);
    manager.persist(c1);
    manager.persist(c2);
    transaction.begin();
    transaction.commit();
  }
}
```

81. **Code-34 to Update the data in @ManyToMany Annotation ?**

    ▼ **Ans**

82. **Code-35 to Fetch the data in @ManyToMany Annotation ?**

    ▼ **Ans**

83. **Code-36 to Delete the data Using @ManyToMany Annotation ?**

    ▼ **Ans**

84. **Bi-Directional Mapping ?**

▼ **Ans**

@JoinColumn

- **It is a field level Annotation present in JPA.**
- **It is used to specify the details of join column (foreign keys).**
- **It is used to specify the owner of foreign keys.**

mappedBy

- **It is an attribute present in OneToOne, OneToMany and ManyToMany. It is used to represent the non-owner of the foreign keys.**

85. **@OneToOne Bi-Directional ?**

▼ **Ans**

86. **Code-34 to Save the data Using @OneToOne (Bi-Direc) Annotation ?**

▼ **Ans**

▼ **persistence.xml**

- **change only Database name —> "One_To_One_Bi?createDatabaseIfNotExist=true"**

▼ **Person.java**

```
package orj.jsp;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;
import javax.persistence.Table;

@Entity
@Table (name = "person")
public class Person
{
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  @Column (nullable = false)
  private String name,place;
  @Column (nullable = false, unique = true)
  private long phone;
  @OneToOne
  @JoinColumn
  private PanCard pan;

  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getPlace() {
    return place;
  }
  public void setPlace(String place) {
```

```
    this.place = place;
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
  public PanCard getPan() {
    return pan;
  }
  public void setPan(PanCard pan) {
    this.pan = pan;
  }
}
```

▼ **PanCard.java**

```
package orj.jsp;

import java.time.LocalDate;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;
import javax.persistence.Table;
@Entity
@Table (name = "PanCard")
public class PanCard
{
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  @Column(nullable = false,unique = true)
  private String number;
  @Column(nullable = false)
  private LocalDate dob;
  @Column(nullable = false)
  private String country;
  @OneToOne (mappedBy = "pan")
  private Person person;

  public Person getPerson() {
    return person;
  }
  public void setPerson(Person person) {
    this.person = person;
  }
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getNumber() {
    return number;
  }
  public void setNumber(String number) {
    this.number = number;
  }
  public LocalDate getDob() {
    return dob;
  }
  public void setDob(LocalDate dob) {
    this.dob = dob;
  }
  public String getCountry() {
    return country;
  }
  public void setCountry(String country) {
    this.country = country;
  }
}
```

▼ **SavePerson.java**

```
package orj.jsp;

import java.time.LocalDate;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class SavePerson
{
  public static void main(String[] args)
  {
    Person p =new Person();
    p.setName("Proffsor");
    p.setPlace("Tokyo");
    p.setPhone(123455);

    PanCard card =new PanCard();
    card.setDob(LocalDate.parse("1997-01-22"));
    card.setNumber("HD543JdHCSK");
    card.setCountry("Japen");
    card.setPerson(p);
    p.setPan(card);

    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    EntityTransaction transaction = manager.getTransaction();

    manager.persist(p);
    manager.persist(card);
    transaction.begin();
    transaction.commit();
  }
}
```

87. **Code-35 to Update the data Using @OneToOne (Bi-Direc) Annotation ?**

▼ **Ans**

88. **Code-36 to Fetch the data Using @OneToOne (Bi-Direc) Annotation ?**

▼ **Ans**

- **We can Fetch in 2 ways**
- Fetching the person name by using Pan ID.

▼ **Using find() method**

```
package orj.jsp;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class FetchPerson2
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();

    PanCard pan = manager.find(PanCard.class, 1);
    if(pan!=null)
    {
      Person p = pan.getPerson();
      System.out.println(p.getName());
      System.out.println(p.getPhone());
      System.out.println(p.getPlace());
    }
    else
```

```
    {
      System.err.println("Invaild ID");
    }
  }
}
--------------OUTPUT-----------
Proffsor
123455
Tokyo
```

▼ **Using HQL Queries**

```
package orj.jsp;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;

public class FetchPerson
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();

    Query q = manager.createQuery("select p from Person p where p.pan.id=?1");
    q.setParameter(1, 1);
    Person p =(Person) q.getResultList().get(0);
    System.out.println(p.getName());
    System.out.println(p.getPhone());
    System.out.println(p.getPlace());
  }
}
------------OUTPUT----------
Proffsor
123455
Tokyo
```

89. **Code-37 to Delete the data Using @OneToOne (Bi-Direc) Annotation ?**

▼ **Ans**

90. **@OneToMany Bi-Directional ?**

▼ **Ans**

- **In Bi-Direction, There is no separate Annotations for OneToMany and ManyToOne. Combination Both called as OneToMany.**

91. **Code-38 to Save the data Using @OneToMany (Bi-Direc) Annotation ?**

▼ **Ans**

▼ **persistence.xml**

- **change only Database name —> "One_To_Many_Bi?createDatabaseIfNotExist=true"**

▼ **Hospital.java**

```
package org.jsp;

import java.util.List;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
@Entity
public class Hospital
{
```

```
  @Id
  @GeneratedValue (strategy = GenerationType.IDENTITY)
  private int id;
  private String name, Gst, founder;
  @OneToMany (mappedBy = "hospital")
  private List<Branch> branchs;

  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getGst() {
    return Gst;
  }
  public void setGst(String gst) {
    Gst = gst;
  }
  public String getFounder() {
    return founder;
  }
  public void setFounder(String founder) {
    this.founder = founder;
  }
  public List<Branch> getBranchs() {
    return branchs;
  }
  public void setBranchs(List<Branch> branchs) {
    this.branchs = branchs;
  }
}
```

▼ **Branch.java**

```
package org.jsp;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
@Entity
public class Branch
{
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  private String name, email;
  private long phone;
  @ManyToOne @JoinColumn(name="hospital_id")
  private Hospital hospital;
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getEmail() {
    return email;
  }
  public void setEmail(String email) {
    this.email = email;
  }
  public long getPhone() {
```

```
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
  public Hospital getHospital() {
    return hospital;
  }
  public void setHospital(Hospital hospital) {
    this.hospital = hospital;
  }
}
```

▼ **SaveHospital.java**

```
package org.jsp;

import java.util.ArrayList;
import java.util.Arrays;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class SaveHospital
{
  public static void main(String[] args)
  {
        Hospital hospital = new Hospital();
        hospital.setName("Oppp");
        hospital.setFounder("Elon musk");
        hospital.setGst("GVVGV684HJJH");

        Branch b1, b2, b3;
        b1 = new Branch();
        b1.setName("AB");
        b1.setEmail("MNUVCHGNBHNM<@gmail.com");
        b1.setPhone(120);
        b1.setHospital(hospital);

        b2 = new Branch();
        b2.setName("ABc");
        b2.setEmail("MNM<@gmail.com");
        b2.setPhone(1111111111);
        b2.setHospital(hospital);

        b3 = new Branch();
        b3.setName("ABkkkk");
        b3.setEmail("popopo@gmail.com");
        b3.setPhone(999999);
        b3.setHospital(hospital);


        hospital.setBranchs(new ArrayList<Branch>(Arrays.asList(b1,b2,b3)));

        EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
        EntityManager manager = factory.createEntityManager();
        EntityTransaction transaction = manager.getTransaction();

        manager.persist(hospital);
        manager.persist(b1);
        manager.persist(b2);
        manager.persist(b3);
        transaction.begin();
        transaction.commit();
  }
}
```

92. **Code-39 to Update the data in @OneToMany (Bi-Direc) Annotation ?**

    ▼ **Ans**

93. **Code-40 to Fetch branches details by Hospital ID by Using Scanner class in @OneToMany (Bi-Direc) Annotation ?**

▼ Ans

- We all know that, We can Fetch the data in 2 ways

▼ FindBranchesByHospitalID.java (Using find() method)

```
package org.jsp;

import java.util.List;
import java.util.Scanner;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

public class FindBranchesByHospitalID
{
  public static void main(String[] args)
  {
    Scanner sc = new Scanner (System.in);
    System.out.println("Enter the Hospital ID:- ");
    int hid = sc.nextInt();
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    Hospital hospital = manager.find(Hospital.class, hid);
    if(hospital!=null)
    {
      List<Branch> branchs = hospital.getBranchs();
      for(Branch b : branchs)
      {
        System.out.println("ID "+b.getId());
        System.out.println("Name "+b.getName());
        System.out.println("Phone "+b.getPhone());
        System.out.println("Email "+b.getEmail());
        System.out.println("---------------------");
      }
    }
    else
    {
      System.err.println("Invaild Hospital ID");
    }

  }
}
```

▼ FindBranchesByHospitalID2.java (Using HQL Queries)

```
package org.jsp;

import java.util.List;
import java.util.Scanner;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;

public class FindBranchesByHospitalID2
{
  public static void main(String[] args)
  {
    Scanner sc = new Scanner (System.in);
    System.out.println("Enter the Hospital ID:- ");
    int hid = sc.nextInt();
    sc.close();
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    Query q = manager.createQuery("select b from Branch b where b.hospital.id=?1");
    q.setParameter(1, hid);
    List<Branch> branchs = q.getResultList();
    if(branchs.size()!=0)
    {
      for(Branch b : branchs)
      {
        System.out.println("ID "+b.getId());
        System.out.println("Name "+b.getName());
```

```
        System.out.println("Phone "+b.getPhone());
        System.out.println("Email "+b.getEmail());
        System.out.println("---------------------");
      }
    }
    else
    {
      System.err.println("Invaild Hospital ID");
    }
  }
}
```

94. **Code-42 to Fetch branches Details by Branch Phone number or Branch Name by Using Scanner class in @OneToMany (Bi-Direc) Annotation ?**

▼ **Ans**

▼ **FetchbranchesByBranchPhoneNumber.java**

```
package org.jsp;

import java.util.List;
import java.util.Scanner;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;

public class FetchbranchesByBranchPhoneNumber {
  public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the Branch Phone Number:- ");
    long phnum = sc.nextLong();
    sc.close();
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    Query q = manager.createQuery("select b from Branch b where b.phone=?1");
    q.setParameter(1, phnum);
    List<Branch> branchs = q.getResultList();
    if (branchs.size() != 0) {
      for (Branch b : branchs) {
        System.out.println("ID " + b.getId());
        System.out.println("Name " + b.getName());
        System.out.println("Phone " + b.getPhone());
        System.out.println("Email " + b.getEmail());
        System.out.println("---------------------");
      }
    } else {
      System.err.println("Invaild Branch Phone Number");
    }
  }
}
```

▼ **FetchbranchesByBranchName.java**

```
package org.jsp;

import java.util.List;
import java.util.Scanner;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;

public class FetchbranchesByBranchName
{
  public static void main(String[] args)
  {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the Branch Name:- ");
    String name = sc.next();
    sc.close();
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
```

```
    Query q = manager.createQuery("select b from Branch b where b.name=?1");
    q.setParameter(1, name);
    List<Branch> branchs = q.getResultList();
    if (branchs.size() != 0)
    {
      for (Branch b : branchs)
      {
        System.out.println("ID " + b.getId());
        System.out.println("Name " + b.getName());
        System.out.println("Phone " + b.getPhone());
        System.out.println("Email " + b.getEmail());
        System.out.println("---------------------");
      }
    }
    else
    {
      System.err.println("Invaild Branch Name");
    }
  }
}
```

95. **Code-40 to Fetch Hospital Details by Hospital Name by Using Scanner class in @OneToMany (Bi-Direc) Annotation ?**

▼ **Ans**

▼ **FetchHospitalDetailsByHospitalName.java**

```
package org.jsp;

import java.util.List;
import java.util.Scanner;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.Query;

public class FetchHospitalDetailsByHospitalName
{
  public static void main(String[] args)
  {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the Hospital Name:- ");
    String name = sc.next();
    sc.close();
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    Query q = manager.createQuery("select h from Hospital h where h.name=?1");
    q.setParameter(1, name);
    Hospital hospital = (Hospital) q.getResultList().get(0);
    System.out.println("ID " + hospital.getId());
    System.out.println("Name " + hospital.getName());
    System.out.println("GST " + hospital.getGst());
    System.out.println("Founder " + hospital.getFounder());
  }
}
```

96. **Code-41 to Fetch Hospital Details by Branch Name by Using Scanner class in @OneToMany (Bi-Direc) Annotation ?**

▼ **Ans**

97. **Code-43 to Delete the data in @OneToMany (Bi-Direc) Annotation ?**

▼ Ans

98. **@ManyToMany Bi-Directional ?**

▼ **Ans**

- **It is a field level Annotation present in JPA.**
- **It is present in javax.persistence package.**

- **It is used to specify the details about join table.**
- **Following are the Attribute of @JoinTable**
  1. **Name**
  2. **JoinColumns**
  3. **Inverse**

99. **Code-44 to Save the data Using @ManyToMany (Bi-Direc) Annotation ?**

▼ **Ans**

▼ **persistence.xml**

- **change only Database name —> "Many_To_Many_Bi?createDatabaseIfNotExist=true"**

▼ **Student.java**

```
package org.jsp;

import java.util.List;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinColumns;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;

@Entity
public class Student
{
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  private String name;
  private long phone;
  private double perc;
  @ManyToMany
  @JoinTable(name = "Student_Course",joinColumns = {@JoinColumn(name="Student_id")},inverseJoinColumns = {@JoinColumn(name="Co
  private List<Course> courses;
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
  public double getPerc() {
    return perc;
  }
  public void setPerc(double perc) {
    this.perc = perc;
  }
  public List<Course> getCourses() {
    return courses;
  }
  public void setCourses(List<Course> courses) {
    this.courses = courses;
  }
}
```

▼ **Course.java**

```
package org.jsp;

import java.util.List;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToMany;

@Entity
public class Course {
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  private String subject;
  private int duration;
  @ManyToMany(mappedBy = "courses")
  private List<Student> students;
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getSubject() {
    return subject;
  }
  public void setSubject(String subject) {
    this.subject = subject;
  }
  public int getDuration() {
    return duration;
  }
  public void setDuration(int duration) {
    this.duration = duration;
  }
  public List<Student> getStudents() {
    return students;
  }
  public void setStudents(List<Student> students) {
    this.students = students;
  }
}
```

▼ **SaveStudent.java**

```
package org.jsp;

import java.util.ArrayList;
import java.util.Arrays;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;

public class SaveStudent
{
  public static void main(String[] args)
  {
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    EntityTransaction transaction = manager.getTransaction();

    Student student1 = new Student();
    student1.setName("A");
    student1.setPhone(1233);
    student1.setPerc(63.36);

    Student student2 = new Student();
    student2.setName("B");
    student2.setPhone(84684);
    student2.setPerc(52.52);
```

```
    Student student3 = new Student();
    student3.setName("C");
    student3.setPhone(12385873);
    student3.setPerc(28.25);

    Course c1 = new Course();
    c1.setSubject("Java");
    c1.setDuration(62);

    Course c2 = new Course();
    c2.setSubject("J2EE");
    c2.setDuration(50);

    Course c3 = new Course();
    c3.setSubject("React JS");
    c3.setDuration(32);

    //Assigning Course for Student
    student1.setCourses(new ArrayList<Course>(Arrays.asList(c1,c2,c3)));
    student2.setCourses(new ArrayList<Course>(Arrays.asList(c1,c2,c3)));
    student3.setCourses(new ArrayList<Course>(Arrays.asList(c2,c3)));

    //Assigning Student for Courser
    c1.setStudents(new ArrayList<Student>(Arrays.asList(student1,student2)));
    c2.setStudents(new ArrayList<Student>(Arrays.asList(student1,student2,student3)));
    c3.setStudents(new ArrayList<Student>(Arrays.asList(student1,student2,student3)));

    manager.persist(student1);
    manager.persist(student2);
    manager.persist(student3);
    manager.persist(c1);
    manager.persist(c2);
    manager.persist(c3);

    transaction.begin();
    transaction.commit();
  }
}
```

00. **Code-45 to Update the data Using @ManyToMany (Bi-Direc) Annotation ?**

    ▼ **Ans**

01. **Code-46 to Fetch the data Using @ManyToMany (Bi-Direc) Annotation ?**

    ▼ **Ans**

02. **Code-47 to Delete the data Using @ManyToMany (Bi-Direc) Annotation ?**

    ▼ **Ans**

---

03. **What is Cascade ?**

    ▼ **Ans**

    - **It is used to modify the state of child of object. Whenever the state of parent object is modify.**
    - **We can specify the cascading type by using an Enum “CascadeType”**
    - **Following are the difference Cascade types**

    1. **CascadeType.PERSIST**
        - **It is used to save the child Entity whenever the parent entity is saved.**
    2. **CascadeType.MERGE**
        - **If we use this, child entity gets updated whenever we update the parent entity.**
    3. **CascadeType.Remove**

- If we use this, child entity get deleted whenever we delete the parent entity.

4. CascadeType.REFRESH
   - If we use this, child entity get refresh whenever we refresh the parent entity.
5. CascadeType.DETACH
   - If we use this, child entity get detach whenever we detach the parent entity.
6. CascadeType.ALL
   - If we use this, any operation on parent entity will affect the child entity.

Note: "It is used for simplify the code, if we use this Cascade Annotation. No need to persist the child class"

04. **Fetch ?**

▼ Ans

- Fetch is an attribute present in @OneToOne, @OneToMany, @ManyToOne @ManyToMany.
- We can specify the fetch type using an Enum called FetchType.
- Following are the different FetchTypes.
  1. FetchType.EAGER
  2. FetchType.LAZY

Mapping	Default
@OneToOne	EAGER
@OneToMany	LAZY
@ManyToOne	EAGER
@ManyToMany	LAZY

05. **Composite Key ?**

▼ Ans

- The Combination of more then one column to form a primary key is called as Composite Key.
- If we make the combination of more then one column as a primary key then their combination in each record must be unique.

Ex: Let us consider the combination of Phone Number and Email ID as Primary Key

Phone	Email ID	
888	a@gmaul.com	Accepted
888	n@gmail.com	Accepted
888	n@gmail.com	Not-Accepted
999	n@gmail.com	Accepted

06. **@Embedable Annotation ?**

▼ Ans

- It is a class-level Annotation present in JPA Present in Javax.persistence package.
- It is used to create composite keys.
- An Embeddable class must implements Serializable Interface.

07. **@Embededd Annotation ?**

▼ Ans

- It is field level Annotation belongs to JPA and present in javax.persistence package.

- It is present in javax.persistence package.
- It is used to represents the Composite key.

08. Code

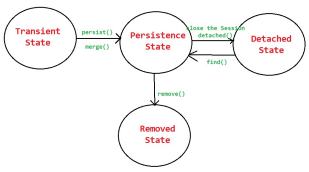
▼ Ans

09. **What is Hibernate Life Cycle ?**

▼ Ans

- Hibernate Life Cycle depicts the different states of an Entity in its life time following are the different states of an object in its life cycle.

1. **Transient State.**
   - An object is said to be in Transient stage as soon as it created.
   - A Transient object will not represent any record.
   - Modification to the Transient object will not reflect in Data-Base.
2. **Persistence State.**
   - An object is said to be present in Persistence state, whenever it is connected with Session or EntityManager.
   - A Persistence object represents a record in the table.
   - Any modification to the persistence object will reflect in the Data-base.
3. **Detached State.**
   - An object is said to be present in Detached state, If it is disconnected from the Session or If we close the Session.
   - An object in detached state is disconnected from the Session, But it will be present in the Data-Base.
   - Any modification on the detached entity will not reflect in the Data-Base.
   - We cannot remove a detached entity.
   - An object will move to the detached state, if we call **detach()** method.
   - We can move an object from detached state to Persistence state by calling **find()** method
4. **Removed State.**
   - An object is said to be in Removed State, If it is deleted from the Data-Base.
   - Object in Removed State is still connected with the Session.



- We cannot delete a record in Detached State.
- We should delete a record in Persistence State.

10. **Code for Hibernate Life Cycle ?**

▼ Ans

▼ **persistence.xml**

- **change only Database name —> “Hibernate_Life_Cycle?createDatabaseIfNotExist=true”**

▼ **User.java**

```
package org.jsp;

public class User
{
 private int id;
 private String name,email;
 private long phone;
public int getId() {
  return id;
}
public void setId(int id) {
  this.id = id;
}
public String getName() {
  return name;
}
public void setName(String name) {
  this.name = name;
}
public String getEmail() {
  return email;
}
public void setEmail(String email) {
  this.email = email;
}
public long getPhone() {
  return phone;
}
public void setPhone(long phone) {
  this.phone = phone;
}

}
```

▼ **SaveUser.java**

```
package org.jsp;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import javax.print.attribute.standard.Media;

public class SaveUser
{
  public static void main(String[] args)
  {
    User u =new User(); //Transient State
    u.setId(2);
    u.setName("Abc");
    u.setEmail("ABC@gmail.com");
    u.setPhone(123);
    EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
    EntityManager manager = factory.createEntityManager();
    EntityTransaction transaction = manager.getTransaction();

    manager.merge(u); //Persistence State
    u.setName("P");
    u.setEmail("P@gmail.com");
    transaction.begin();
    transaction.commit();

    manager.detach(u);//Detached State
    u.setName("R");
    u.setEmail("R@gmail.com");
    u = manager.find(User.class, 2);//Persistence State
    manager.remove(u);   //Removed state
    transaction.begin();
    transaction.commit();
```

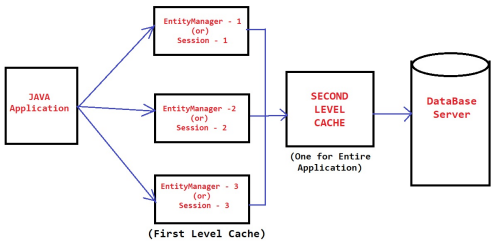
```
    }
}
```

11. **What is Cache ?**

    ▼ **Ans**

    - **It is layer of storage which is used to store the data to serve the future requests faster. Hibernate supports 2 level of cache**

    1. **First Level Cache**
        - **Cache memory is separate for each Session or EntityManager.**
        - **An Application can have any number of first level Cache which depends upon number of Sessions or EntityManagers.**
        - **By default, Hibernate supports First Level Cache.**
    2. **Second Level Cache**
        - **Every First level Cache will be connected to Second Level Cache.**
        - **Second Level Cache is one for the entire application.**
        - **If we want Second Level Cache in our application then we must have a dependency.**
        - **To enable second level cache in application, We should add a dependency “Hibernate EhCache Relocation” —> 5.6.5**



12. **Code for cache ?**

    ▼ **Ans**

13. **All Annotation ?**

    ▼ **Ans**

    1. **@Entity**
    2. **@Id**
    3. **@GeneratedValue (strategy)**
    4. **@Table (name)**
    5. **@Column (name, unique, nullable)**
    6. **@JoinColumn (name)**
    7. **@Cacheable**
    8. **@OneToOne (mapped by , Fetch Cascade)**
    9. **@OneToMany (mapped by , Fetch Cascade)**
    10. **@ManyToOne (Fetch Cascade)**
    11. **@ManyToMany (mapped by , Fetch Cascade)**

12. **@JoinTable (name, joinColumn, inverse JoinColumn)**
13. **@Embeddable**
14. **@Embededd**
15. **CascadeType**
16. **FetchType**
17. **GenerationType**

---

1. **Modularization :**
   a. **dto (Used to store the Entity Class)**
   b. **dao (Data Access Object)**
   c. **Servlets (Used to store the Servlets)**
2. **Code for Modularization or How to achieve Modularization ?**

▼ **org.jsp.UserApp.controller**

▼ **SaveUser.java**

```
package org.jsp.UserApp.controller;

import java.util.Scanner;

import org.jsp.UserApp.dao.UserDao;
import org.jsp.UserApp.dto.User;

public class SaveUser
{//ABC,888,s@gmail.com,s123
  public static void main(String[] args)
  {
    Scanner sc = new Scanner(System.in);

    System.out.println("Enter the Name to Save : ");
    String name = sc.next();

    System.out.println("Enter the Email to Save : ");
    String email = sc.next();

    System.out.println("Enter the Phone Number to Save : ");
    long phone = sc.nextLong();

    System.out.println("Enter the Password to Save : ");
    String pass = sc.next();
    sc.close();

    User u = new User();
    u.setName(name);
    u.setPhone(phone);
    u.setEmail(email);
    u.setPassword(pass);
    UserDao dao = new UserDao();
    u = dao.SaveUser(u);
    System.out.println("User Saved with ID "+u.getId());
  }
}
```

▼ **UpdateUser.java**

```
package org.jsp.UserApp.controller;

import java.util.Scanner;

import org.jsp.UserApp.dao.UserDao;
import org.jsp.UserApp.dto.User;

public class UpdateUser
{
```

```
  public static void main(String[] args)
  {
    Scanner sc = new Scanner(System.in);

    System.out.println("Enter the ID to Update");
    int id = sc.nextInt();

    System.out.println("Enter the Name to Update : ");
    String name = sc.next();

    System.out.println("Enter the Email to Update: ");
    String email = sc.next();

    System.out.println("Enter the Phone Number to Update : ");
    long phone = sc.nextLong();

    System.out.println("Enter the Password to Update : ");
    String pass = sc.next();
    sc.close();

    User u = new User();
    u.setId(id);
    u.setName(name);
    u.setPhone(phone);
    u.setEmail(email);
    u.setPassword(pass);
    UserDao dao = new UserDao();
    u = dao.UpdateUser(u);
    System.out.println("User Updated");
  }
}
```

▼ **FetchUserByID.java**

```
package org.jsp.UserApp.controller;

import java.util.Scanner;

import org.jsp.UserApp.dao.UserDao;
import org.jsp.UserApp.dto.User;

public class FetchUserByID
{
  public static void main(String[] args)
  {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the ID to Fetch : ");
    int id = sc.nextInt();
    sc.close();

    UserDao dao = new UserDao();
    User u = dao.findUserById(id);
    if(u!=null)
    {
      System.out.println("Name : "+u.getName());
      System.out.println("Email : "+u.getEmail());
      System.out.println("Phone : "+u.getPhone());
      System.out.println("Password : "+u.getPassword());
    }
    else
    {
      System.out.println("Invaild ID");
    }
  }
}
```

▼ **FetchAllUser.java**

```
package org.jsp.UserApp.controller;

import java.util.List;

import org.jsp.UserApp.dao.UserDao;
import org.jsp.UserApp.dto.User;
```

```
public class FetchAllUser
{
  public static void main(String[] args)
  {
    UserDao dao = new UserDao();
    List<User>  user = dao.FindAllUsers();
    if(user.size()!=0)
    {
      for(User u : user)
      {
        System.out.println("ID : "+u.getId());
        System.out.println("Name : "+u.getName());
        System.out.println("Email : "+u.getEmail());
        System.out.println("Password : "+u.getPassword());
        System.out.println("Phone : "+u.getPhone());
        System.out.println("------------------");
      }
    }
    else
    {
      System.out.println("No Records");
    }
  }
}
```

▼ **DeleteSave.java**

```
package org.jsp.UserApp.controller;

import java.util.Scanner;

import org.jsp.UserApp.dao.UserDao;

public class DeleteSave
{
  public static void main(String[] args)
  {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter the ID to Delete : ");
    int id = sc.nextInt();
    sc.close();

    UserDao dao = new UserDao();
    boolean u = dao.DeleteUser(id);
    if(u)
      System.out.println(id+" is Deleted");
    else
      System.out.println("Invalid ID");
  }
}
```

▼ **VerifyUser.java**

```
package org.jsp.UserApp.controller;

import java.util.Scanner;

import org.jsp.UserApp.dao.UserDao;
import org.jsp.UserApp.dto.User;

public class VerifyUser
{
  public static void main(String[] args)
  {
    Scanner sc = new Scanner(System.in);

    System.out.println("Enter the phone : ");
    long phone = sc.nextLong();

    System.out.println("Enter the password : ");
    String password = sc.next();
    sc.close();

    UserDao dao = new UserDao();
    User u = dao.VerifyUser(phone, password);
```

```
    if(u!=null)
    {
      System.out.println(u.getId());
      System.out.println(u.getName());
      System.out.println(u.getEmail());
    }
    else
    {
      System.out.println("Invalid Phone or Password");
    }
  }
}
```

▼ **org.jsp.UserApp.dao**

▼ **UserDao.java**

```
package org.jsp.UserApp.dao;

import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import javax.persistence.Query;

import org.jsp.UserApp.dto.User;

public class UserDao
{
  EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
  EntityManager manager = factory.createEntityManager();

  public User SaveUser(User User)
  {
    EntityTransaction transaction = manager.getTransaction();
    manager.persist(User);
    transaction.begin();
    transaction.commit();
    return User;
  }
  public User UpdateUser(User User)
  {
    EntityTransaction transaction = manager.getTransaction();
    manager.merge(User);
    transaction.begin();
    transaction.commit();
    return User;
  }
  public boolean DeleteUser(int id)
  {
    EntityTransaction transaction = manager.getTransaction();
    User u = manager.find(User.class, id);
    if(u!=null)
    {
      manager.remove(u);
      transaction.begin();
      transaction.commit();
      return true;
    }
    return false;
  }
  public User findUserById(int id)
  {
    return manager.find(User.class, id);
  }
  public List<User> FindAllUsers()
  {
    Query q = manager.createQuery("select u from User u");
    return q.getResultList();
  }
  public User VerifyUser(long phone , String password)
  {
    Query q = manager.createQuery("select u from User u where u.phone=?1 and u.password=2?");
    q.setParameter(1, phone);
    q.setParameter(2, password);
    List <User> user = q.getResultList();
    if(user.size()>0)
```

```
      return user.get(0);
    return null;
  }
}
```

▼ **org.jsp.UserApp.dto**

▼ **User.java**

```
package org.jsp.Serv_HiberApp.dto;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class User
{
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  private String name, email, password;
  private long phone;

  public int getId() {
    return id;
  }

  public void setId(int id) {
    this.id = id;
  }

  public String getName() {
    return name;
  }

  public void setName(String name) {
    this.name = name;
  }

  public String getEmail() {
    return email;
  }

  public void setEmail(String email) {
    this.email = email;
  }

  public String getPassword() {
    return password;
  }

  public void setPassword(String password) {
    this.password = password;
  }

  public long getPhone() {
    return phone;
  }

  public void setPhone(long phone) {
    this.phone = phone;
  }
}
```

3. **JSP (Java Server Page) ?**

▼ **Ans**

- **It is an extension of servlet which is used for development of dynamic web application, following are the different objects present in JSP.**

1. **request → It is reference variable of HttpServletRequest.**

2. response → It is a reference variable of HttpServletResponse.
3. Session → It is a reference variable of HttpSession.
4. out → It is a reference variable of JSPWriter.

- We can write the Java code in JSP with the help of following tags.
  1. Declaration Tag <%! %>
     - It is used to declare the variables and define the methods.
  2. Scriplet tag <% %>
     - This tag is used to write the Logic.
  3. Expression tag <%= %>
     - It is used to print an Expression or Value.

4. **Servlet Chaining ?**

▼ Ans

- The process of transferring the control from one Resource to another Resource such as Servlet, JSP or HTML is called Servlet Chaining.
- We can achieve Servlet Chaining with the help of RequestDispature and ServletResponse both are Interfaces.

1. RequestDispature
   a. forward (ServletRequest , ServletResponse)
   b. include (ServletRequest , ServletResponse)
2. ServletResponse
   a. sendRedirect (String)

5. **Code to Create Login Validation using Servlet Chaining ?** (Normal Servlet Project)

▼ Ans

▼ web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http:/
  <display-name>LoginAndRegistrationUsingServletChaining</display-name>
  <welcome-file-list>
    <welcome-file>login.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

▼ login.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
  <form action="login" method="post">
    <input type="tel" name="ph">
    <input type="password" name="ps">
    <input type="submit" value="login">
  </form>
</body>
</html>
```

▼ **LoginServlet.java**

```
package org.jsp;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/login")
public class LoginServlet extends HttpServlet
{
  @Override
  protected void doPost(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException
  {
    long phone = Long.parseLong(req.getParameter("ph"));
    String password = req.getParameter("ps");
    RequestDispatcher dispatcher = null;
    PrintWriter writer = resp.getWriter();

    if(phone == 123456 && password.equals("a123"))
    {
      dispatcher = req.getRequestDispatcher("home.jsp");
      dispatcher.forward(req, resp);
    }
    else
    {
      writer.write("<html><body><h1>Invalid Credentials</h1></body></html>");
      dispatcher = req.getRequestDispatcher("login.jsp");
      dispatcher.include(req, resp);
      //resp.sendRedirect("");
    }
  }
}
```

▼ **home.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
  <h1>Login Successful</h1>
  <a>Add User</a>
  <a>Update User</a>
  <a>Delete User</a>
  <a>Fetch User</a>
</body>
</html>
```

6. **Steps to Create a Web Application using Maven ? (Login+Registration+CRUD) Project Using Servlet and Hibernate ?**

▼ **Ans**

Project Set-Up :-

1. **Create a Maven Project by selecting the following Archetype** (It is a not a Simple Project —> Don't select Simple Maven Project)
   a. **Group Id → "org.apache.maven.archetypes"**
   b. **Artifact Id → "maven-archetype-webapp"**
   c. **Version → "1.4"** (Up-to this step is enough to get version 1.4, If you doesn't get, then add below Repository URL Link)

d. **Repository URL → "https://repo.maven.apache.org/maven2/archetype-catalog.xml"**

- **Select org.apache.maven.archetypes → maven-archetype-webapp → 1.4 then click on Next.**
- **Then Give Group Id and Artifact Id —> as based on Project.**

2. **Go to Deployed Resource Folder in our Project → webapp Folder → (delete index.html) File.**
3. **We will get src folder, inside src we should have (main→java) and (test→java) folder. In case, If we don't find them. We have one solution. Right Click on Our Project —> Select build path** (If we don't get build path, don't worry go to properties from there we will get Java Build Path) **—> In Build Path Go to order and Export —> Select JRE System Library and Maven Dependencies then Click on Apply and Close. —> Now, We can see (main→java) and (test→java) folder.**
4. **If we want to use JPA, We need to create a one configuration file, that file is persistence.xml. We should create persistence.xml in META-INF and META-INF is present inside src/main/resource folder. But, In our Project there is no src/main/resource folder. So, That we need create META-INF manually inside src→main→webapp→WEB-INF →(create a folder called "classes") → (Inside classes create another folder called "META-INF") → (Inside META-INF create "persistence.xml")**
5. **Add the following dependencies into the pom.xml**
   a. **Hibernate Core Relocation from maven repository (5.6.5) —> "**https://mvnrepository.com/artifact/org.hibernate/hibernate-core/5.6.5.Final**"**
   b. **MySQL Connector Java from maven repository (8.0.28) —> "**https://mvnrepository.com/artifact/mysql/mysql-connector-java/8.0.28**"**
   c. **Servlet api (4.0.1) → "**https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api/4.0.1**"**
   - **As soon as we save it, We will get Maven Dependencies folder. We can check that in Java Resource Folder — Inside—> Libraries Folder.**

Project Development :-

- **We should Create 3 packages inside src/main/java folder which is present inside Java Resources folder**

▼ **org.jsp.UserWebApp.dto**

▼ **User.java**

```
package org.jsp.UserWebApp.dto;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class User
{
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  private String name, email, password;
  @Column (nullable = false, unique = true)
  private long phone;

  public int getId() {
    return id;
  }

  public void setId(int id) {
    this.id = id;
  }

  public String getName() {
    return name;
  }

  public void setName(String name) {
```

```
    this.name = name;
  }

  public String getEmail() {
    return email;
  }

  public void setEmail(String email) {
    this.email = email;
  }

  public String getPassword() {
    return password;
  }

  public void setPassword(String password) {
    this.password = password;
  }

  public long getPhone() {
    return phone;
  }

  public void setPhone(long phone) {
    this.phone = phone;
  }
}
```

▼ **org.jsp.UserWebApp.dao**

▼ **UserDao.java**

```
package org.jsp.UserWebApp.dao;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import javax.persistence.Query;

import org.jsp.UserWebApp.dto.User;


public class UserDao
{
  EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
  EntityManager manager = factory.createEntityManager();

  public User SaveUser(User User)
  {
    EntityTransaction transaction = manager.getTransaction();
    manager.persist(User);
    transaction.begin();
    transaction.commit();
    return User;
  }
  public User UpdateUser(User User)
  {
    EntityTransaction transaction = manager.getTransaction();
    manager.merge(User);
    transaction.begin();
    transaction.commit();
    return User;
  }
  public boolean DeleteUser(int id)
  {
    EntityTransaction transaction = manager.getTransaction();
    User u = manager.find(User.class, id);
    if(u!=null)
    {
      manager.remove(u);
      transaction.begin();
      transaction.commit();
      return true;
    }
    return false;
  }
  public User findUserById(int id)
```

```
  {
    return manager.find(User.class, id);
  }
  public List<User> FindAllUsers()
  {
    Query q = manager.createQuery("select u from User u");
    return q.getResultList();
  }
  public User VerifyUser(long phone , String password)
  {
    Query q = manager.createQuery("select u from User u where u.phone=?1 and u.password=?2");
    q.setParameter(1, phone);
    q.setParameter(2, password);
    List<User> user = q.getResultList();
    if(user.size()>0)
      return user.get(0);
    return null;
  }
}
```

▼ **org.jsp.UserWebApp.controller**

▼ **LoginServlet.java**

```
package org.jsp.UserWebApp.controller;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import org.jsp.UserWebApp.dao.UserDao;
import org.jsp.UserWebApp.dto.User;

@WebServlet("/login")
public class LoginServlet extends HttpServlet
{
  @Override
  protected void doPost(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException
  {
    long phone = Long.parseLong(req.getParameter("ph"));
    String password = req.getParameter("ps");
    UserDao dao = new UserDao();
    User u = dao.VerifyUser(phone, password);
    RequestDispatcher dispatcher = null;
    PrintWriter writer = resp.getWriter();
    if(u!=null)
    {
      HttpSession session = req.getSession();
      session.setAttribute("user", u);
      dispatcher = req.getRequestDispatcher("home.jsp");
      dispatcher.forward(req, resp);
      writer.close();
    }
    else
    {
      writer.write("<html><body><h1>Invalid Credentials</h1></body></html>");
      dispatcher = req.getRequestDispatcher("login.jsp");
      dispatcher.include(req, resp);
      writer.close();
    }
  }
}
```

▼ **SignUpServlet.java**

```
package org.jsp.UserWebApp.controller;

import java.io.IOException;
```

```
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.jsp.UserWebApp.dao.UserDao;
import org.jsp.UserWebApp.dto.User;

@WebServlet("/signup")
public class SignUpServlet extends HttpServlet
{
  @Override
  protected void doPost(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException
  {
    String name = req.getParameter("nm");
    String password = req.getParameter("ps");
    String email = req.getParameter("em");
    long phone = Long.parseLong(req.getParameter("ph"));
    User u = new User();
    u.setEmail(email);
    u.setPassword(password);
    u.setPhone(phone);
    u.setName(name);
    UserDao dao = new UserDao();
    u = dao.SaveUser(u);
    PrintWriter writer = resp.getWriter();
    writer.write("<html><body><h1>User Saved with ID:"+u.getId()+"</h1></body></html>");
    writer.close();
  }
}
```

▼ **EdituserServlet.java**

```
package org.jsp.UserWebApp.controller;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.jsp.UserWebApp.dao.UserDao;
import org.jsp.UserWebApp.dto.User;

@WebServlet("/edit")
public class EdituserServlet extends HttpServlet
{
  @Override
  protected void doPost(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException
  {
    int id = Integer.parseInt(req.getParameter("id"));
    String name = req.getParameter("nm");
    String password = req.getParameter("ps");
    String email = req.getParameter("em");
    long phone = Long.parseLong(req.getParameter("ph"));
    User u = new User();
    u.setId(id);
    u.setEmail(email);
    u.setPassword(password);
    u.setPhone(phone);
    u.setName(name);
    UserDao dao = new UserDao();
    u = dao.UpdateUser(u);
    PrintWriter writer = resp.getWriter();
    writer.write("<html><body><h1>User Saved with ID:"+u.getId()+"</h1></body></html>");
    writer.close();
  }
}
```

▼ **DeleteServlet.java**

```
package org.jsp.UserWebApp.controller;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

import org.jsp.UserWebApp.dao.UserDao;

@WebServlet("/delete")
public class DeleteServlet extends HttpServlet
{
  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException
  {
    int id = Integer.parseInt(req.getParameter("id"));
    UserDao dao = new UserDao();
    dao.DeleteUser(id);
    HttpSession session = req.getSession();
    session.invalidate();
    PrintWriter writer = resp.getWriter();
    writer.write("<html><body><h1>Account Deleted</h1></body></html>");
    RequestDispatcher dispatcher = req.getRequestDispatcher("login.jsp");
    dispatcher.include(req, resp);
    writer.close();
  }
}
```

▼ **LogoutServlet.java**

```
package org.jsp.UserWebApp.controller;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

@WebServlet("/logout")
public class LogoutServlet extends HttpServlet
{
  @Override
  protected void doGet(HttpServletRequest req, HttpServletResponse resp)
      throws ServletException, IOException
  {
    HttpSession session = req.getSession();
    session.invalidate();
    RequestDispatcher dispatcher = req.getRequestDispatcher("login.jsp");
    dispatcher.forward(req, resp);
  }
}
```

- **We should Create JSP Resources inside webapp folder which is present inside main folder which is present inside src folder which is present inside our project.**

▼ **login.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
```

```
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
  <form action="login" method="post">
    Phone<input type="tel" name="ph"><br>
    PAssword<input type="password" name="ps"><br>
    <input type="submit" value="LOGIN">
  </form>
  <a href="signup.jsp">Sign Up</a>
</body>
</html>
```

▼ **signup.jsp**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
  <form action="signup" method="post">
    Name:<input type="text" name="nm"><br>
    Phone:<input type="tel" name="ph"><br>
    Email:<input type="email" name="em"><br>
    Password:<input type="password" name="ps"><br>
    <input type="submit" value="Signup">
  </form>
</body>
</html>
```

▼ **home.jsp**

```
<%@page import="org.jsp.UserWebApp.dto.User"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
  <%User u = (User) session.getAttribute("user");
  if(u!=null)
  {%>
  <h1>Login Succesfull</h1>
  <h2><a href="edit.jsp">Click Here to Edit your Details</a></h2>
  <h2><a href="delete?id=<%=u.getId()%>">Click Here to Delete your Details</a></h2>
  <h2><a href="view.jsp">Click Here to view your Details</a></h2>
  <h2><a href="logout">Click Here to Logout</a></h2>
  <%}
  else
  {
    response.sendRedirect("login.jsp");
  }%>
</body>
</html>
```

▼ **view.jsp**

```
<%@page import="org.jsp.UserWebApp.dto.User"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
  pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
```

```
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
  <%User u = (User) session.getAttribute("user");
  if(u!=null)
  {%>
  <h1>Name:<%=u.getName() %></h1>
  <h1>EMail:<%=u.getEmail() %></h1>
  <%} %>
</body>
</html>
```

▼ **edit.jsp**

```
<%@page import="org.jsp.UserWebApp.dto.User"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%User u = (User) session.getAttribute("user");
  if(u!=null)
  {%>
  <form action="edit" method="post">
  <input type="number" name="id" value="<%=u.getId() %>" readonly="readonly"><br>
  <input type="text" name="nm" value="<%=u.getName() %>"><br>
  <input type="tel" name="ph" value="<%=u.getPhone() %>"><br>
  <input type="email" name="em" value="<%=u.getEmail() %>"><br>
  <input type="password" name="ps" value="<%=u.getPassword() %>"><br>
  <input type="submit"value="Update"><br>
  </form>
  <%}
  else
  {
    response.sendRedirect("login.jsp");
  }
  %>
</body>
</html>
```

▼ **pom.xml** (pom.xml present inside Our Project only)

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>org.jsp.UserWebApp</groupId>
  <artifactId>User-Web-App</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>war</packaging>

  <name>User-Web-App Maven Webapp</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>

  <dependencies>
  <!-- https://mvnrepository.com/artifact/org.hibernate/hibernate-core -->
<dependency>
    <groupId>org.hibernate</groupId>
    <artifactId>hibernate-core</artifactId>
    <version>5.6.5.Final</version>
</dependency>
```

```
  <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.28</version>
</dependency>
  <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>4.0.1</version>
    <scope>provided</scope>
</dependency>

    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <finalName>User-Web-App</finalName>
    <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults (may be moved to parent pom) -->
      <plugins>
        <plugin>
          <artifactId>maven-clean-plugin</artifactId>
          <version>3.1.0</version>
        </plugin>
        <!-- see http://maven.apache.org/ref/current/maven-core/default-bindings.html#Plugin_bindings_for_war_packaging -->
        <plugin>
          <artifactId>maven-resources-plugin</artifactId>
          <version>3.0.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.8.0</version>
        </plugin>
        <plugin>
          <artifactId>maven-surefire-plugin</artifactId>
          <version>2.22.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-war-plugin</artifactId>
          <version>3.2.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-install-plugin</artifactId>
          <version>2.5.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-deploy-plugin</artifactId>
          <version>2.8.2</version>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>
```

▼ **web.xml** (web.xml present inside “WEB-INF” folder which is present inside “webapp” folder which is present inside “main” folder which is present inside “src” folder)

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >

<web-app>
  <display-name>Archetype Created Web Application</display-name>
  <welcome-file-list>
  <welcome-file>login.jsp</welcome-file>
  </welcome-file-list>
</web-app>
```

▼ **persistence.xml** (persistence.xml present inside “META-INF” folder which is present inside “classes” folder which is present inside “WEB-INF” folder which is present inside “webapp” folder present inside “main” folder which is present inside “src” folder)

```
<persistence xmlns="http://xmlns.jcp.org/xml/ns/persistence"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
  http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd"
  version="2.1">

  <persistence-unit name="dev">
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>
    <properties>
      <property name="javax.persistence.jdbc.driver"
        value="com.mysql.cj.jdbc.Driver" />
      <property name="javax.persistence.jdbc.url"
        value="jdbc:mysql://localhost:3306/User-Web-App?createDatabaseIfNotExist=true"/>
      <property name="javax.persistence.jdbc.user" value="root" />
      <property name="javax.persistence.jdbc.password"
        value="admin" />
      <property name="hibernate.show_sql" value="true" />

      <property name="hibernate.hbm2ddl.auto" value="update" />
      <property name="hibernate.dialect"
        value="org.hibernate.dialect.MySQL8Dialect" />
    </properties>
  </persistence-unit>
</persistence>
```

7. **Hospital App Project ? (Last day of Hibernate class) ?**

▼ **Ans**

- **This Normal Simple Maven Project**

▼ **pom.xml**

- **Add dependencies Hibernate core relocation and SQL connector**

▼ **persistence.xml**

- **Change DataBase Name**

▼ **org.jsp.HospitalApp.dto**

▼ **Hospital.java**

```
package org.jsp.HospitalApp.dto;
import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToMany;
@Entity
public class Hospital
{
  @Id
  @GeneratedValue (strategy = GenerationType.IDENTITY)
  private int id;
  private String name, Gst, founder;
  @OneToMany (mappedBy = "hospital" , cascade = CascadeType.ALL)
  private List<Branch> branchs;
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
```

```
  public void setName(String name) {
    this.name = name;
  }
  public String getGst() {
    return Gst;
  }
  public void setGst(String gst) {
    Gst = gst;
  }
  public String getFounder() {
    return founder;
  }
  public void setFounder(String founder) {
    this.founder = founder;
  }
  public List<Branch> getBranchs() {
    return branchs;
  }
  public void setBranchs(List<Branch> branchs) {
    this.branchs = branchs;
  }


}
```

▼ **Branch.java**

```
package org.jsp.HospitalApp.dto;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
@Entity
public class Branch
{
  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;
  private String name, email;
  private long phone;
  @ManyToOne @JoinColumn(name="hospital_id")
  private Hospital hospital;
  public int getId() {
    return id;
  }
  public void setId(int id) {
    this.id = id;
  }
  public String getName() {
    return name;
  }
  public void setName(String name) {
    this.name = name;
  }
  public String getEmail() {
    return email;
  }
  public void setEmail(String email) {
    this.email = email;
  }
  public long getPhone() {
    return phone;
  }
  public void setPhone(long phone) {
    this.phone = phone;
  }
  public Hospital getHospital() {
    return hospital;
  }
  public void setHospital(Hospital hospital) {
    this.hospital = hospital;
  }
}
```

▼ **org.jsp.HospitalApp.dao**

▼ **HospitalDao.java**

```
package org.jsp.HospitalApp.dao;

import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import javax.persistence.Query;
import org.jsp.HospitalApp.dto.Hospital;

public class HospitalDao {
  EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
  EntityManager manager = factory.createEntityManager();

  public Hospital saveHospital(Hospital hospital) {
    EntityTransaction transaction = manager.getTransaction();
    manager.persist(hospital);
    transaction.begin();
    transaction.commit();
    return hospital;
  }

  public Hospital getHospital(int id) {
    return manager.find(Hospital.class, id);
  }

  public Hospital updateHospital(Hospital hospital) {
    EntityTransaction transaction = manager.getTransaction();
    manager.merge(hospital);
    transaction.begin();
    transaction.commit();
    return hospital;
  }

  public void deleteHospital(int id) {
    Hospital hospital = manager.find(Hospital.class, id);
    if (hospital != null) {
      EntityTransaction transaction = manager.getTransaction();
      manager.remove(hospital);
      transaction.begin();
      transaction.commit();
    }
  }

  @SuppressWarnings("unchecked")
  public List<Hospital> getAllHospital() {
    String qry = "select h from Hospital h";
    Query query = manager.createQuery(qry);
    List<Hospital> hospitals = query.getResultList();
    return hospitals;
  }
}
```

▼ **BranchDao.java**

```
package org.jsp.HospitalApp.dao;

import java.util.List;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityTransaction;
import javax.persistence.Persistence;
import javax.persistence.Query;

import org.jsp.HospitalApp.dto.Branch;
import org.jsp.HospitalApp.dto.Hospital;

public class BranchDao {
  EntityManagerFactory factory = Persistence.createEntityManagerFactory("dev");
  EntityManager manager = factory.createEntityManager();
```

```
  public Branch saveBranch(Branch branch, int hospital_id) {
    Hospital hospital = manager.find(Hospital.class, hospital_id);
    if (hospital != null) {
      EntityTransaction transaction = manager.getTransaction();
      hospital.getBranchs().add(branch); // setting branch to hospital
      branch.setHospital(hospital);// setting hospital to branch
      manager.persist(branch);
      transaction.begin();
      transaction.commit();
      return branch;
    }
    return null;
  }

  public Branch updateBranch(Branch branch, int hospital_id) {
    Hospital hospital = manager.find(Hospital.class, hospital_id);
    if (hospital != null) {
      EntityTransaction transaction = manager.getTransaction();
      hospital.getBranchs().add(branch); // setting branch to hospital
      branch.setHospital(hospital);// setting hospital to branch
      manager.merge(branch);
      transaction.begin();
      transaction.commit();
      return branch;
    }
    return null;
  }

  public Branch getBranchById(int id) {
    return manager.find(Branch.class, id);
  }

  public List<Branch> getBranchByHospitalId(int id) {
    String query = "select b from Branch b where b.hospital.id=?1";
    Query q = manager.createQuery(query);
    q.setParameter(1, id);
    List<Branch> branchs = q.getResultList();
    return branchs;
  }

  public void deleteBranch(int id) {
    Branch branch = manager.find(Branch.class, id);
    if(branch!=null)
    {
      EntityTransaction transaction = manager.getTransaction();
      manager.remove(branch);
      transaction.begin();
      transaction.commit();
    }
  }
}
```

▼ org.jsp.HospitalApp.controller

▼ SaveHospital.java

```
package org.jsp.HospitalApp.controller;

import org.jsp.HospitalApp.dao.HospitalDao;
import org.jsp.HospitalApp.dto.Hospital;

public class SaveHospital
{
  public static void main(String[] args)
  {
    Hospital hospital = new Hospital();
    hospital.setName("Apollo");
    hospital.setFounder("Founder");
    hospital.setGst("A123");
    HospitalDao dao = new HospitalDao();
    hospital = dao.saveHospital(hospital);
    System.out.println("Hospital Saved with ID "+hospital.getId());
  }
}
```

▼ SaveBranch.java

```
package org.jsp.HospitalApp.controller;

import org.jsp.HospitalApp.dao.BranchDao;
import org.jsp.HospitalApp.dto.Branch;

public class SaveBranch
{
  public static void main(String[] args)
  {
    Branch branch = new Branch();
    branch.setEmail("ABC123@gmaiil.com");
    branch.setName("Apollo Bengalore");
    branch.setPhone(12345678);

    BranchDao dao = new BranchDao();
    branch = dao.saveBranch(branch, 1);
    if(branch!=null)
      System.out.println("Branch is Saved with ID "+branch.getId());
    else
      System.out.println("Hospital Not Present");
  }
}
```

▼ **GetBranchByHospital.java**

```
package org.jsp.HospitalApp.controller;

import java.util.List;

import org.jsp.HospitalApp.dao.BranchDao;
import org.jsp.HospitalApp.dto.Branch;

public class GetBranchByHospital
{
  public static void main(String[] args)
  {
    BranchDao dao = new BranchDao();
    List<Branch> branchs = dao.getBranchByHospitalId(1);
    for(Branch b : branchs)
    {
      System.out.println("ID : "+b.getId());
      System.out.println("Name : "+b.getName());
      System.out.println("Phone : "+b.getPhone());
      System.out.println("Email : "+b.getEmail());
      System.out.println("-------------------");
    }
  }
}
```