

<b>ACCESO A DATOS</b>	<b>PRÁCTICA HIBERNATE</b>	<b>2 DAM</b>
-----------------------	-------------------------------	--------------

## ÍNDICE

Base de datos:

La BD debe constar de al menos tres tablas relacionadas entre sí.

2.1 Deben existir relaciones 1:N

2.2 Debe existir algún campo autonumérico.

2.3 Tipo enumerado en alguna tabla

2.4 Deben poder realizarse operaciones de inserción, consultar, modificar, eliminar en todas las tablas de la BD

Menú de cada clase.

3.1 Clase cliente

3.2 Clase coche

3.3 Clase revisión

Utilización del patrón DAO.

Gestión de transacciones

Opciones no obligatorias

6.2 Tipo fecha en alguna tabla, con consultas donde se maneje este tipo.

Consultas que se realizan contra la BD, indicando si son HQL consultas nativas.

Opciones de cascade que se han configurado, explicando la causa de elección.

Dificultades encontradas en el desarrollo.

9.1. A la hora de añadir los JARs.

9.2. Llamar al toString()

9.3. Sentencia query

Decisiones de optimización o cualquier otra decisión de desarrollo o aclaración que interese destacar.

Dos versiones una grafica con Java FX y otra por consola.

Fallos a corregir

Consulta a de agrupación.

SOLUCIÓN

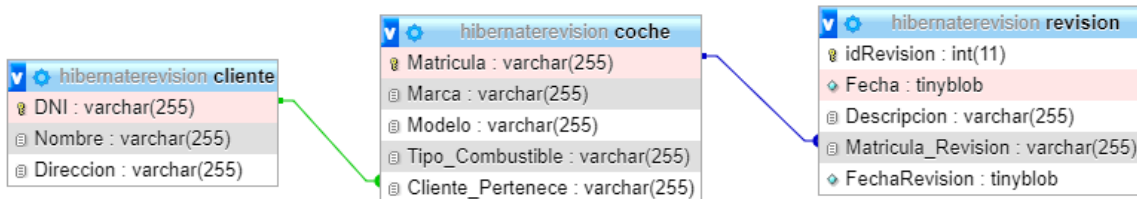
Borrar revision que no borre todas la revision que muestre todas las id de la revision de esa matricula y solicite el id.

SOLUCIÓN.

Borrar el coche que tiene revision.

SOLUCIÓN

## 1. Base de datos:



## 2. La BD debe constar de al menos tres tablas relacionadas entre sí.

### 2.1 Deben existir relaciones 1:N

Todas tiene relaciones 1:N

### 2.2 Debe existir algún campo autonumérico.

En la clase Revisión el idRevisión es autonumerico.

### 2.3 Tipo enumerado en alguna tabla

La tabla Coche tiene el tipo enumerado que se llama Tipo\_Combustible el cual tiene tres tipo de combustible un coche  
ELECTRICO | DIESEL | GASOLINA

### 2.4 Deben poder realizarse operaciones de inserción, consultar, modificar, eliminar en todas las tablas de la BD

En cada una de las tabla se puede insertar datos.

En cada una de las tabla se puede consultar datos.

En cada una de las tabla se puede modificar datos.

En cada una de las tabla se puede eliminar datos.

<b>ACCESO A DATOS</b>	<b>PRÁCTICA HIBERNATE</b>	<b>2 DAM</b>
-----------------------	-------------------------------	--------------

### 3. Menú de cada clase.

#### 3.1 Clase cliente

```
System.out.println("|1| Dar de alta a un cliente.");
System.out.println("|2| Modificar el nombre de un cliente.");
System.out.println("|3| Borrar a un cliente por el dni.");
System.out.println("|4| Consultar Cliente por un nombre.");
System.out.println("|5| Consultar todos los clientes.");
System.out.println("|6| Salir.");
```

#### 3.2 Clase coche

```
System.out.println("|1| Dar de alta a un Coche.");
System.out.println("|2| Modificar modelo de un coche.");
System.out.println("|3| Borrar coche por su matricula.");
System.out.println("|4| Consultar lista de revision de un coche por matricula.");
System.out.println("|5| Consultar todos los Coches.");
System.out.println("|6| Salir.");
```




#### 3.3 Clase revisión

```
System.out.println("|1| Dar de alta a una revision.");
System.out.println("|2| Modificar fecha de una revision.");
System.out.println("|3| Borrar revision por su id.");
System.out.println("|4| Consultar revision de un cliente.");
System.out.println("|5| Consultar todas las revisiones.");
System.out.println("|6| Salir.");
```

<b>ACCESO A DATOS</b>	<b>PRÁCTICA HIBERNATE</b>	<b>2 DAM</b>
-----------------------	-------------------------------	--------------

## 4.Utilización del patrón DAO.

Cada clase tiene su patrón DAO que todas extiendes de la clase GenericDAO()

 ClienteDao.java	 CocheDao.java	 RevisionDao.java
<p>Tiene 3 método Cliente buscarClientePorDni(String)</p> <p>List&lt;Cliente&gt; consultarClientesPorNombre(String)</p> <p>List&lt;Cliente&gt; consultarTodosLosClientes()</p>	<p>Tiene 3 método void cambiarMatricula (Coche ,String)</p> <p>Coche buscarCochePorMatricula (String)</p> <p>List&lt;Coche&gt; consultarTodosLosCoche s()</p>	<p>Tiene 3 método Revision consultarRevisionPorid(in t)</p> <p>Cliente buscarClientePorDni(Strin g)</p> <p>List&lt;Revision&gt; consultarTodasRevisione s()</p>

## 5. Gestión de transacciones

En la clase GenericDAO

Utilización de beginTransaction() y getTransaction

```
public class GenericDAO<T> {  
    public void guardar(T entidad) {  
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();  
        session.beginTransaction();  
        session.save(entidad);  
        session.getTransaction().commit();  
    }  
    public void borrar(T entidad) throws RevisionException{  
        Session session = HibernateUtil.getSessionFactory().getCurrentSession();  
        session.beginTransaction();  
        session.delete(entidad);  
        session.getTransaction().commit();  
    }  
}
```

## 6. Opciones no obligatorias

6.2 Tipo fecha en alguna tabla, con consultas donde se maneje este tipo.

En la clase revision hay una variable de tipo LocalDate fechaRevision

```
@Column(name = "FechaRevision")  
private LocalDate fechaRevision;
```

Consulta

1. Cliente
2. Coche
3. Revision
4. Salir

- 1| Dar de alta a una revision.
- 2| Modificar fecha de una revision.
- 3| Borrar revision por su id.
- 4| Consultar revision de un cliente.
- 5| Consultar todas las revisiones.
- 6| Salir.

Donde se pedirá el id y la matrícula del coche

En el caso que el id que se introduzca no exista en la base de datos. Aparezca el siguiente mensaje

```
|2| Modificar fecha de una revision.
Ingrese el id de la revision que cambiara la fecha:
5
Ingresa la matricula del coche:
1258MPK
Hibernate: select coche0_.Matricula as Matricula1_, co
Hibernate: select revision0_.idRevision as idRevision2_
Vaya. Parece que no existe el id 5
```

Y por último también si la matrícula introducida no existe en la base de datos. Aparece el siguiente mensaje.

```
Ingresa la matricula del coche:
1234
Hibernate: select coche0_.Matricula as Matricula1_, coche0_.Cli
Vaya. Parece que no existe ese coche con esa matricula 1234
```

En el caso de que todos los datos sean correctos

```
Revision [idRevision=4, fechaRevision=2021-02-02, descripcion=Presion de los neumaticos, coche=Coche
matricula=1258MPK
marca=Seat
modelo=Leon
cliente=Cliente
dni=29658978Y
    nombre=Manuel
    direccion=Calle conde
    listCoches=1

Tipo combustible ELECTRICO
listRevisiones= 2
-
Procedemos a cambiar la fecha
Ingresa el dia [NUEVA FECHA]
16
Ingresa el mes [NUEVA FECHA]
02
Ingresa el año [NUEVA FECHA]
2021
Hibernate: update Revision set Matricula_Revision=?, Descripcion=?, FechaRevision=? where idRevision=?
Fecha de la revision cambiada correctamente.

Revision [idRevision=4, fechaRevision=2021-02-16, descripcion=Presion de los neumaticos, coche=Coche
matricula=1258MPK
marca=Seat
modelo=Leon
cliente=Cliente
dni=29658978Y
    nombre=Manuel
    direccion=Calle conde
    listCoches=1

Tipo combustible ELECTRICO
listRevisiones= 2
1
```

Los cambios se modifican correctamente.

ACCESO A DATOS	PRÁCTICA HIBERNATE	2 DAM
----------------	-----------------------	-------

## 7. Consultas que se realizan contra la BD, indicando si son HQL consultas nativas.

Todas las consultas que se han realizado son **HQL**.

## 8. Opciones de cascade que se han configurado, explicando la causa de elección.

### Clase Cliente

```
@OneToMany(cascade = CascadeType.ALL)
@JoinColumn(name = "Cliente_Pertenece")
private List<Coche> listCoches;
```

**CascadeType.ALL** para cuando algo se modifica. En la listCoche también será modificado o eliminado.

### Clase Coche

```
@ManyToOne(fetch = FetchType.LAZY)
@Cascade(org.hibernate.annotations.CascadeType.REFRESH)
@JoinColumn(name = "Cliente_Pertenece")
```

**CascadeType.SAVE\_REFRESH** para cuando se borre o se modifique un coche no se borre ni modifique el cliente.

```
@OneToMany
@Cascade(org.hibernate.annotations.CascadeType.ALL)
@JoinColumn(name = "Matricula_Revision")
```

**CascadeType.ALL** para cuando modifique o borre un coche, la lista de revisiones se modifique o se borre

### Clase Revision

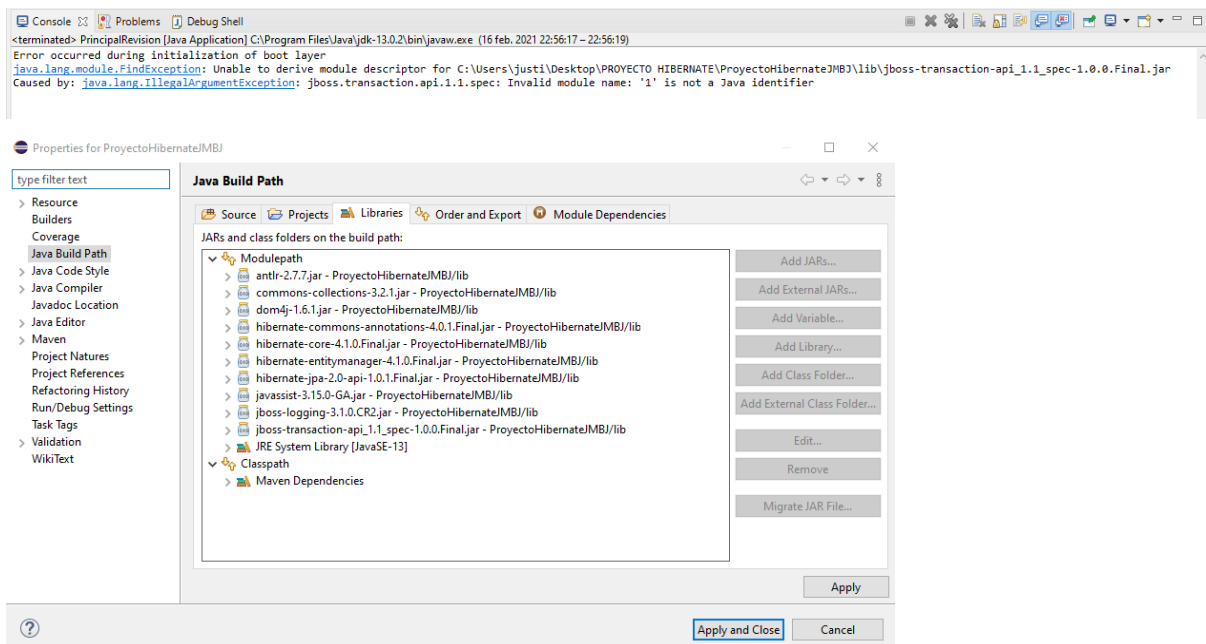
```
@ManyToOne()  
@Cascade(org.hibernate.annotations.CascadeType.SAVE_UPDATE)  
@JoinColumn(name = "Matricula_Revision")  
private Coche coche;
```

en este caso he decidido ponerle **CascadeType.SAVE\_UPDATE** para cuando se borre o se modifique una revisión no se borre ni modifique el coche.

## 9. Dificultades encontradas en el desarrollo.

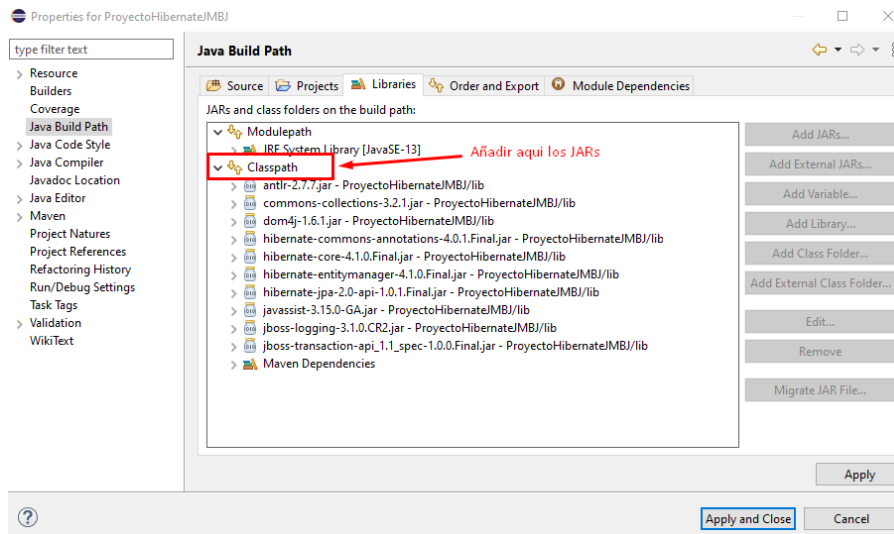
### 9.1. A la hora de añadir los JARs.

Lo estaba agregando en el Modulepath  
y me aparecía lo siguiente



Para solucionar este problema fue necesario borrar del modulepath y agregarlo donde tenían que estar los JARs.





## 9.2. Llamar al toString()

al hacer una consulta a todos los clientes.

```
[5] Consultar todos los clientes.
hibernate: select cliente0_.DNI as DNI0_, cliente0_.Direccion as Direccion0_, cliente0_.Nombre as Nombre0_ from Cliente cliente0_
hibernate: select listcoches0_.Cliente_Pertenece as Cliente4_0_1_, listcoches0_.Matricula as Matricula1_, listcoches0_.Matricula
Exception in thread "main" java.lang.StackOverflowError
    at java.base/java.lang.String.valueOf(String.java:3352)
    at java.base/java.lang.StringBuilder.append(StringBuilder.java:166)
    at java.base/java.util.AbstractCollection.toString(AbstractCollection.java:457)
    at org.hibernate.collection.internal.PersistentBag.toString(PersistentBag.java:501)
    at java.base/java.lang.String.valueOf(String.java:3352)
    at java.base/java.lang.StringBuilder.append(StringBuilder.java:166)
    at clases.Cliente.toString(Client.java:127)
    at java.base/java.lang.String.valueOf(String.java:3352)
    at java.base/java.lang.StringBuilder.append(StringBuilder.java:166)
    at clases.Coche.toString(Coche.java:129)
```

descubrí que no tenía el tamaño .size();

```
@Override
public String toString() {

    return "Cliente \ndni=" + this.dni + ""
        + "\n\tnombre=" + this.nombre + ""
        + "\n\tdireccion=" + this.direccion + ""
        + "\n\tlistCoches=" + this.listCoches
        + "\n";
}
```

Una vez encontrado el fallo el método toString() funciona correctamente.

Clase Cliente

```

@Override
public String toString() {
    return "Cliente \ndni=" + this.dni + ""
        + "\n\tnombre=" + this.nombre + ""
        + "\n\tdireccion=" + this.direccion + ""
        + "\n\tlistCoches=" + this.listCoches.size()
        + "\n";
}

```

Para la clase coche

```

@Override
public String toString() {
    return "Coche [matricula=" + matricula + ", marca=" + marca + ", modelo=" + modelo + ", cliente=" + cliente
        + ", listRevisiones=" + listRevisiones.size() + "];"
}

```

tuve que modificar hibernate.cfg.xml

```

<property name="hibernate.hbm2ddl.auto">update</property>

```

## 9.3. Sentencia query

### 9.3.1 Al realizar esa sentencia no pillaba el \*

```

public static List<Cliente> consultarClientesPorNombre(String nombre){
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    List<Cliente> listClienteNombres;

    Query sentencia = session.createQuery("SELECT * FROM Cliente WHERE Nombre = '"+nombre+"'");
    //Nos devuelve una lista con los nombre que sea igual
    listClienteNombres = sentencia.list();
    return listClienteNombres;
}

```

```

Exception in thread "main" org.hibernate.hql.internal.ast.QuerySyntaxException: unexpected token: * near line 1, column 8 [SELECT * FROM clases.Cliente WHERE Nombre = 'jose']
at org.hibernate.hql.internal.ast.QuerySyntaxException.convert(QuerySyntaxException.java:54)
at org.hibernate.hql.internal.ast.QuerySyntaxException.convert(QuerySyntaxException.java:47)
at org.hibernate.hql.internal.ast.ErrorCounter.throwQueryException(ErrorCounter.java:29)
at org.hibernate.hql.internal.ast.QueryTranslatorImpl.parse(QueryTranslatorImpl.java:276)
at org.hibernate.hql.internal.ast.QueryTranslatorImpl.doCompile(QueryTranslatorImpl.java:188)
at org.hibernate.hql.internal.ast.QueryTranslatorImpl.compile(QueryTranslatorImpl.java:136)

```

Para solucionarlo fue necesario añadir a Cliente un alias "c"

```

public static List<Cliente> consultarClientesPorNombre(String nombre){
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    List<Cliente> listClienteNombres;

    Query sentencia = session.createQuery("SELECT c FROM Cliente c WHERE Nombre = '"+nombre+"'");
    //Nos devuelve una lista con los nombre que sea igual
    listClienteNombres = sentencia.list();
    return listClienteNombres;
}

```

```
|4| Consultar Cliente por un nombre.
Ingrese el nombre del cliente que desea consultar:
antonio
Hibernate: select cliente0_.DNI as DNI0_, cliente0_.Direccion as Direccion0_, cliente0_.Nombre as Nombre0_ from Cliente cliente0_ where Nombre='antonio'
Cliente
dni=30286639M
nombre=antonio
direccion=clipper
listCoche=0
```

Mismo error en la siguiente sentencia y misma solución.

```
public static List<Cliente> consultarTodosLosClientes(){
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    List<Cliente> listClientes;

    Query sentencia = session.createQuery("SELECT c FROM Cliente c");

    listClientes= sentencia.list();
    return listClientes;
}
```

## 10. Decisiones de optimización o cualquier otra decisión de desarrollo o aclaración que interese destacar.

Añadir Tag `<query>`

Existe dos manera de hacerlo

**name:** Este atributo define el nombre de la consulta. Es el nombre que posteriormente usaremos desde el código Java para acceder a la consulta.

**contenido:** El contenido del tag `<query>` es la consulta en formato **HQL** que ejecutará Hibernate.

Primera forma es creando un **XML**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC "-//Hibernate/Hibernate
> <hibernate-mapping>
>   <class name="clases.Cliente">
>       <id column="Id" name="DNI" />
>       <property name="Nombre" />
>       <property name="Direccion" />
>
>   </class>
>
>   <query name="Cliente.findAllClientes"><![CDATA[
>       SELECT c FROM Cliente c
>   ]]></query>
> </hibernate-mapping>
```

Segunda forma es añadirla en la propia clase

```

@Entity()
@Table (name = "Cliente")
@NamedQueries ({
    @NamedQuery ( name = "Cliente.findAll",
                  query = "SELECT c FROM Cliente c")
})
public class Cliente implements Serializable {

```

Posibles Errores

```
name = "Cliente.findAll"
```

En el método no tiene el mismo nombre.

```

|5| Consultar todos los clientes.
Exception in thread "main" org.hibernate.MappingException: Named query not known: findAllClientes
at org.hibernate.internal.AbstractSessionFactory.getNamedQuery(AbstractSessionFactory.java:140)

```

Para utilizarla

1. llama al método `getNamedQuery()` que se le pasa el nombre mediante el tag `<query>`.
2. se ha definido la consulta llamada `findAllClientes`

```

public static List<Cliente> consultarTodosLosClientes(){
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    List<Cliente> listClientes;

    Query sentenciaQuery = session.getNamedQuery("findAllClientes");

    listClientes= sentenciaQuery.list(); 1 2
    return listClientes;
}

```

## Dos versiones una grafica con Java FX y otra por consola.

Lo he realizado en JavaFX para tener dos versiones una seria por consola y la otra mas graficamente. Aquí dejo el link al proyecto.

Lo comparto afuera porque pesa 15MB  
y el de consola casi 10MB

LINK =

[https://drive.google.com/file/d/1ok9w9QmyWlx-JKaMxSbVORnh8Z1W\\_i2y/view?usp=sharing](https://drive.google.com/file/d/1ok9w9QmyWlx-JKaMxSbVORnh8Z1W_i2y/view?usp=sharing)

ACCESO A DATOS	PRÁCTICA HIBERNATE	2 DAM
----------------	-----------------------	-------

## Fallos a corregir

### A) Consulta a de agrupación.

Que lo que hará es decirme cuantos coches

**Electrico** tiene ese cliente.

**Diesel** tiene ese cliente .

**Gasolina** tiene ese cliente.

y lo agrupara por su tipo de combustible.

Añado una consulta mas al menú Coche

```

1| Dar de alta a un Coche.
2| Modificar modelo de un coche.
3| Borrar coche por su matricula.
4| Consultar lista de revision de un coche por matricula.[Ordena por fecha]
5| Consultar todos los Coches.
6| Cuantos coches tiene ese cliente agrupado por el tipo de combustible
7| Salir.

```

Me pedirá el dni del cliente.

Introduce el dni del coche que deseas consultar

```

6| Cuantos coches tiene ese cliente agrupado por el tipo de combustible.
Introduce el dni del coche que deseas consultar

```

me dirá lo siguiente.

```

6| Cuantos coches tiene ese cliente agrupado por el tipo de combustible.
Introduce el dni del coche que deseas consultar
6498653P
Hibernate: select coche0_.Tipo_Combustible as col_0_0_, count(coche0_.Tipo_Con
COMBUSTIBLE DIESEL TIENE 2
COMBUSTIBLE GASOLINA TIENE 1

```

En el caso contrario

```

6| Cuantos coches tiene ese cliente agrupado por el tipo de combus
Introduce el dni del coche que deseas consultar
5659
Hibernate: select coche0_.Tipo_Combustible as col_0_0_, count(coche
Vaya. Parece ser que ese dni: 5659 No tiene ningun coche.

```

## SOLUCIÓN

En el Menú Coche

Añado en el switch la opción 6

```

        case 6:
            System.out.println("|" + opcCoche + "| Cuantos coches tiene ese cliente agrupado por el tipo de combustible.");
            dniCliente = solicitarCadena("Introduce el dni del coche que deseas consultar");
            consultarCochePorDni(dniCliente);

            break;

```

Creó un nuevo método en el principal.

```

private static void consultarCochePorDni(String dniCliente) throws RevisionException {
    List<Object[]> listaCocheDAO = cocheDao.consultarCochePorDNI(dniCliente);

    if (listaCocheDAO.isEmpty()) {
        throw new RevisionException("Vaya. Parece ser que ese dni: " + dniCliente + " No tiene ningun coche.");
    }

    for (Object[] objects : listaCocheDAO) {
        System.out.println("COMBUSTIBLE " + objects[0] + " TIENE " + objects[1]);
    }
}

```

Creó un nuevo método en la clase CocheDao

```

public List<Object[]> consultarCochePorDNI(String dni){
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    List<Object[]> listCocheDni;
    Query sentenciaQuery = session.createQuery("SELECT co.tipoCombustible COUNT(co.tipoCombustible) FROM Coche co WHERE '"+dni+"' = co.cliente GROUP BY co.tipoCombustible");
    listCocheDni = sentenciaQuery.list();
    return listCocheDni;
}

```

Tiene que ser el mismo nombre de la clase Coche para realizar la consulta.

```

private Cliente cliente;

@Column(name = "Tipo_Combustible")
@Enumerated(EnumType.STRING)
private TipoCombustible tipoCombustible;

```

<b>ACCESO A DATOS</b>	<b>PRÁCTICA HIBERNATE</b>	<b>2 DAM</b>
-----------------------	-------------------------------	--------------

B) Borrar revision que no borre todas la revision que muestre todas las id de la revision de esa matricula y solicite el id.

En el apartado de

```

1. Cliente
2. Coche
3. Revision
4. Salir

|1| Dar de alta a una revision.
|2| Modificar fecha de una revision.
|3| Borrar revision por su matricula.
|4| Consultar revision de un cliente.
|5| Consultar todas las revisiones.
|6| Salir.

```

```

|3| Borrar revision por su matricula.
Ingrese la matricula del coche:
7122

```

Borraremos la revision con la matricula **7122**

```

Id revision:22
Matricula 7122
Descripcion Cambio de frenos trasero
Fecha de la revision:2021-05-06
Id revision:23
Matricula 7122
Descripcion Cambio de aceite
Fecha de la revision:2021-02-09
Ingrese un id:

```

1. Id de la revision esa matricula
2. Introducimos el id de la revision que deseas borrar

Si introduce una id que no pertenece a esa matrícula saltará la excepción.

```

|3| Borrar revision por su matricula.
Ingrese la matricula del coche:
7122
Hibernate: select revision0_.idRevision as idRevision2_, revision0_.Matricula_Revision as
Hibernate: select coche0_.Matricula as Matricula1_0_, coche0_.Cliente_Pertenece as Cliente
Id revision:22
Matricula 7122
Descripcion Cambio de frenos trasero
Fecha de la revision:2021-05-06
Id revision:23
Matricula 7122
Descripcion Cambio de aceite
Fecha de la revision:2021-02-09
Ingrese un id:
20
Vaya. Parece que el id no es correcto 20

```

1. No has ingresado ninguno de los dos
2. Por eso salta la excepción que es incorrecto.

ACCESO A DATOS	PRÁCTICA HIBERNATE	2 DAM
----------------	-----------------------	-------

En el caso contrario de haber introducido el 21

```

Hibernate: select from Revision where Matricula = 21
Id revision:22
Matricula 7122
Nombre de la revision: Cambio de frenos trasero
Fecha de la revision:2021-05-06
Revision con Matricula_Revision 7122 borrada correctamente.

```

Borrará esa revision.

## SOLUCIÓN.

En el **RevisionDao**

```

/**
 * public List<Revision> consultarRevisionPorMatricula (String matricula){
 *     Session session = HibernateUtil.getSessionFactory().getCurrentSession();
 *
 *     List<Revision> listRevisionesFechaMatricula;
 *
 *     Query sentenciaQuery = session.createQuery("SELECT r FROM Revision r WHERE Matricula_Revision = '"+matricula+"'");
 *
 *     listRevisionesFechaMatricula = sentenciaQuery.list();
 *
 *     return listRevisionesFechaMatricula;
 * }
 */

```

cambio la consulta.

→ Principal

```

case 3:
    do {
        opcRevision = verMenuRevision();
        tratarMenuRevision(opcRevision);
    } while (opcRevision != 6);
    break;
}

```



→ tratarMenuRevision 3

```

case 3:

    System.out.println("|" + opcRevision + "| Borrar revision por su matricula.");
    matricula = solicitarCadena("Ingrese la matricula del coche:");
    borrarLaRevisionPorSuMatricula(matricula);
    break;

}

private static void borrarLaRevisionPorSuMatricula(String matricula) throws RevisionException {

    List<Revision> listRevisiones = revisionDao.consultarRevisionPorMatricula(matricula);
    boolean borrar = false;
    int idRevision;
    if (listRevisiones.size() == 0) {
        throw new RevisionException("Vaya. No hay revisiones con esa matricula");
    }
    // Mostrar los id de la revision por la matricula y la fecha de revision
    System.out.println("Lista de revision de la matricula: " + matricula);
    Iterator<Revision> it = listRevisiones.iterator();
    while (it.hasNext()) {
        Revision revision = (Revision) it.next();
        System.out.println("Id revision:" + revision.getIdRevision());
        System.out.println("Matricula " + revision.getCoche().getMatricula());
        System.out.println("Descripcion " + revision.getDescripcion());
        System.out.println("Fecha de la revision:" + revision.getFechaRevision());
    }
    /**
     * solicito el id de la revision
     */
    idRevision = solicitarEntero("Ingrese un id:");

    for (Revision revision : listRevisiones) {
        if (revision.getIdRevision() == idRevision) {
            revision.getCoche().deleteRevision(revision);
            revisionDao.borrar(revision);
            borrar = true;
        }
    }

    if (borrar) {
        System.out.println("Revision con id : " + idRevision + " borrada correctamente.");
    } else {
        throw new RevisionException("Vaya. Parece que no hay ninguna revision con ese id: " + idRevision);
    }
}

```

1. Creó un boolean para saber si se ha borrado la revision.
2. Muestra las revisiones de esa matrícula con los datos IdRevision, Matricula, Descripción y Fecha de la revision.
3. Solicito el id que se va a borrar
4. Hago un For Each diciendo si la revision con ese id es igual a la que previamente se ingresó. Si es esa la borra si no es pasará a la siguiente en el caso que tenga más de dos revisiones dicha matrícula.
5. Si se ha borrado mostrará un mensaje diciendo que se ha borrado. En caso contrario mostrará la excepción diciendo que no se encontró ese idRevision.

C)Borrar el coche que tiene revision.

```
1. Cliente
2. Coche
3. Revision
4. Salir
2
|1| Dar de alta a un Coche.
|2| Modificar modelo de un coche.
|3| Borrar coche por su matricula.
|4| Consultar lista de revision de un coche por matricula.[Ordena por fecha]
|5| Consultar todos los Coches.
|6| Salir.
3
```

```
----- Sistema Operativo -----
|3| Borrar coche por su matricula.
Ingrese la matricula del coche que se va a borrar:
```

```
7122
```

```
Hibernate: select coche0, Matricula as Matricula12 from
```

Coche

```
matricula=7122
marca=Nissan
modelo=L
```

Cliente

```
dni=123456789K
nombre=Justiniano
direccion=Conde
Numero de Coches=0
```

Tipo combustible

```
DIESEL
```

```
listRevisiones= 2
```

## SOLUCIÓN

```
@ManyToOne(fetch = FetchType.LAZY)
@Cascade(org.hibernate.annotations.CascadeType.SAVE_UPDATE)
@JoinColumn(name = "Matricula_Revision")
private Coche coche;
```

Añado el **FetchType.LAZY**