```java
public class ArchivoConfiguracion {

    public static void main(String[] args) throws FileNotFoundException, IOException {

        // Crear un archivo de configuración a partir de datos que introduzca el usuario,
        // y leer este archivo a continuación

        // Leer la entrada del usuario desde consola
        Scanner teclado = new Scanner(System.in);
        // Crear un objeto Properties llamado archivoConfig para almacenar las propiedades de config. del programa
        // La clase Properties se utiliza para manejar listas de valores clave
        Properties archivoConfig = new Properties();

        // Pedir al usuario que introduzca lo que sea,
        System.out.println("Introduce la web: ");
        // se guarda en una variable,
        String web = teclado.nextLine();
        // y se mete en el objeto archivoConfig a través de .setProperty(clave, valor),
        // donde "clave" es el identificador de una propiedad y "valor" su valor asociado.
        archivoConfig.setProperty("url", web);

        System.out.println("Introduce el prefijo de la tabla: ");
        String prefijo = teclado.nextLine();
        archivoConfig.setProperty("prefix", prefijo);

        System.out.println("Servidor de BBDD: ");
        String server = teclado.nextLine();
        archivoConfig.setProperty("server", server);

        // Las propiedades contenidas en el obejto se guardan en un archivo
        // llamado "wp-config.php", creado a partir del flujo de salida FileOutputStream
        // El método .store() toma dos argumentos: el nombre del archivo y un comentario para conocer su propósito
        archivoConfig.store(new FileOutputStream("wp-config.php"),
                "Mi primer archivo de configuraciÃ³n");

        // Leer contenido de archivoConfig con el método .load(), tomando como argumento
        // el flujo de entrada que representa al archivo "wp-config.php"
        archivoConfig.load(new FileInputStream("wp-config.php"));

        // En una variable String, guardamos el valor de una propiedad obtenida del objeto,
        // con el método .getProperty(buscando por clave)
        String servidor = archivoConfig.getProperty("server");
        // Mostramos por pantalla un texto con ese valor concatenado
        System.out.println("Servidor de la BBDD " + servidor);

        String dominio = archivoConfig.getProperty("url");
        System.out.println("El dominio es: " + dominio);
    }
}
```

```java
public class Sentencias_Puras {

    public static void main(String[] args) throws SQLException {

        // Importamos las clases necesarias
        // Cargamos el driver JDBC en el proyecto

        // Identificamos el origen de los datos en la variable cadenaConexion (RUTA)
        String cadenaConexion = "jdbc:mysql://localhost:3306/alumnos";
        // Para gestionar el cierre de recursos se mete el código dentro de un bloque try-with-resources.
        // Más conciso y seguro que un bloque try-catch
        // Creamos un objeto CONEXIÓN, establecemos la conexión con la BD
        try ( Connection conexion = DriverManager.getConnection(cadenaConexion, "root", "1234");
                // Creamos una sentencia sql con .createStatement()
                Statement sentencia = conexion.createStatement();) {
            // Mensaje para el usuario si la conexión se ha establecido correctamente
            System.out.println("Conexión correcta");

            // A partir de aquí utilizamos los métodos .execute(sentenica sql) para trabajar con la BD
            // Crear BD alumnos
            sentencia.execute("create database alumnos");
            System.out.println("Base de datos alumnos creada");
            // Crear tabla con sus campos
            sentencia.execute("CREATE TABLE `alumnos`.`uax` (`id` INT NOT NULL , `nombre` VARCHAR(255) NOT NULL ,"
                    + " `apellidos` VARCHAR(255) NOT NULL , PRIMARY KEY (`id`)) ENGINE = InnoDB;");
            System.out.println("Tabla uax creada correctamente");
            // Para ejecutar instrucciones usamo .executeUpdate(INSERT, DELETE o UPDATE)

            // INSERT
            sentencia.executeUpdate("INSERT INTO `uax` (`id`, `nombre`, `apellidos`) "
                    + "VALUES ('1', 'Martin', 'Scorsese');");
            sentencia.executeUpdate("INSERT INTO `uax` (`id`, `nombre`, `apellidos`) "
                    + "VALUES ('2', 'Ridley', 'Scott');");

            // DELETE
            sentencia.executeUpdate("DELETE FROM uax WHERE `uax`.`id` = 1");
            System.out.println("Dire borrado");

            // UPDATE
            sentencia.executeUpdate("UPDATE `uax` SET `apellidos` = 'Scotteeeeeex' WHERE `uax`.`id` = 2");
            System.out.println("Dire actualizado");

            // SELECT
            // Se declara un objeto ResultSet rs en el que se guardará el resultado de la CONSULTA SQL
            // La consulta se ejecuta con un .executeQuery(sql)
            ResultSet rs = sentencia.executeQuery("SELECT * FROM uax");
            System.out.println("Recorrido en un ResultSet");

            // Variable contador inicializada en 0 para contar los registros
            int contador = 0;
            // En un bucle while SE RECORRE el ResultSet para mostrar los registros obtenidos
            // El método .next() se utiliza para avanzar a la siguiente fila de resultados hasta que se acaben
            while (rs.next()) {
                // Incrementamos el contador
                contador++;
                // Obtener los valores de las columnas de la fila actual
                // Métodos .getInt(int posición) y .getString(int posición)
                // Se almacena el valor del campo id en un entero con el método .getInt()
                int id = rs.getInt("id");
                // Se almacena el valor del campo nombre y apellido en un String con el método .getString()
                String nombre = rs.getString("nombre");
                String apellidos = rs.getString("apellidos");
                // Mensaje concatenado del registro completo
                System.out.println(id + " " + nombre + " " + apellidos);
            }
            // Mensaje de numero de registros
            System.out.println("Hay " + contador + "registros");

        } catch (SQLException ex) {
            // Manejamos excepción si la conexión ha fallado
            System.out.println("Conexión fallida");
            ex.printStackTrace();
        }
    }
}
```

```java
public class Sentencias_Preparadas {

    public static void main(String[] args) throws SQLException {

        // Identificamos el origen de los datos en la variable cadenaConexion (RUTA)
        String cadenaConexion = "jdbc:mysql://localhost:3306/alumnos";
        // Creamos un objeto CONEXIÓN, establecemos la conexión con la BD usando el constructor
        // DriverManager.getConnection(url, user, pass)
        Connection conexion = DriverManager.getConnection(cadenaConexion, "root", "1234");

        // Se definen instrucciones y consultas como cadenas de texto para ser utilizadas
        // en sentencias preparadas, que serán ejecutadas con valores específicos
        String sqlP1 = "INSERT INTO `uax` (`id`, `nombre`, `apellidos`)" // Insertar datos en la tabla
                + "VALUES (?, ?, ?);"; // según posiciones
        String sqlP2 = "DELETE FROM uax WHERE uax.id = ?"; // Borrar datos según la posición del valor
        String sqlP3 = "SELECT * FROM uax WHERE uax.nombre = ?;";

        // Se crea el objeto PreparedStatement a partir de la conexión y una instrucción/consulta
        // Vamos cambiando el parámetro entre sqlP1, sqlP2 y sqlP3
        PreparedStatement sentenciaPrep = conexion.prepareStatement(sqlP3);

        // INTERACCIÓN CON USUARIO
        Scanner teclado = new Scanner(System.in);

        // INSERT
        // Recogida de datos, solicitados al usuario. Se guardan en variables
        System.out.println("Dame el id");
        int id = teclado.nextInt();
        System.out.println("Dame el nombre");
        String nombre = teclado.next();
        System.out.println("Dame el apellido");
        String apellidos = teclado.next();
        // Se muestra el contenido de las variables
        System.out.println(id + " " + nombre + " " + apellidos);
        // Asociar a posiciones los valores recogigos
        // .setInt() añade en la posición el int x
        sentenciaPrep.setInt(1, id);
        // .setString() añade en la posición el String x
        sentenciaPrep.setString(2, nombre);
        sentenciaPrep.setString(3, apellidos);
        // Ejecutar la sentencia preparada con .executeUpdate()
        sentenciaPrep.executeUpdate();
        System.out.println("Datos introducidos correctamente");

        // DELETE
        // Recogida de datos. Se guarda en la variable id
        System.out.println("Dame el id: ");
        String id = teclado.next();
        // Asociar a posiciones los valores recogigos con .setString()
        sentenciaPrep.setString(1, id);
        // Ejecutar la sentencia preparada con .executeUpdate()
        sentenciaPrep.executeUpdate();
        System.out.println("Datos borrados correctamente");

        // SELECT (CONSULTA)
        // Recogida de datos. Ingresar un nombre para realizar una consulta
        System.out.println("Dame el nombre: ");
        String nombre = teclado.next();
        // Asociar el nombre ingresado a la sentenica preparada
        sentenciaPrep.setString(1, nombre);
        // Ejecutar la sentencia preparada con .executeQuery() y guardarlo en un objeto ResultSet
        ResultSet resultadoConsulta = sentenciaPrep.executeQuery();
        System.out.println("Salida de la consulta: ");
        // Recorrer el ResultSet y mostrar el resultado de la consulta por pantalla
        while (resultadoConsulta.next()) {
            int idTabla = resultadoConsulta.getInt(1);
            String nombreTabla = resultadoConsulta.getString(2);
            String apellidoTabla = resultadoConsulta.getString(3);
            System.out.println("id: " + idTabla + " Nombre: " + nombreTabla + " Apellido: " + apellidoTabla);
        }
        System.out.println("Fin del programa");
    }
}
```

```java
public class Consulta_goles {

    public static void main(String[] args) throws SQLException {
        String cadenaConexion = "jdbc:mysql://localhost:3306/equipo";

        try ( Connection conexion = DriverManager.getConnection(cadenaConexion, "root", "1234");
                Statement sentencia = conexion.createStatement();) {
            System.out.println("Conexión correcta");

            sentencia.execute("create database equipo");
            System.out.println("Base de datos alumnos creada");

            sentencia.execute("CREATE TABLE `equipo`.`jugadores` (`id` INT NOT NULL , `nombre` VARCHAR(255) NOT NULL , "
                    + "`apellidos` VARCHAR(255) NOT NULL , `goles` INT, PRIMARY KEY (`id`)) ENGINE = InnoDB");
            System.out.println("Tabla jugadores creada correctamente");

            sentencia.executeUpdate("INSERT INTO `jugadores` (`id`, `nombre`, `apellidos`, `goles`) "
                    + "VALUES ('1', 'Martin', 'Scorsese', '17');");
            sentencia.executeUpdate("INSERT INTO `jugadores` (`id`, `nombre`, `apellidos`, `goles`) "
                    + "VALUES ('2', 'Ridley', 'Scott', '12');");
            sentencia.executeUpdate("INSERT INTO `jugadores` (`id`, `nombre`, `apellidos`, `goles`) "
                    + "VALUES ('3', 'Stanley', 'Kunrick', '20');");
            sentencia.executeUpdate("INSERT INTO `jugadores` (`id`, `nombre`, `apellidos`, `goles`) "
                    + "VALUES ('4', 'Christopher', 'Nolan', '9');");

            ResultSet rs = sentencia.executeQuery("SELECT * FROM jugadores WHERE goles > 12");
            System.out.println("Recorrido en un ResultSet");

            while (rs.next()) {
                int id = rs.getInt("id");
                String nombre = rs.getString("nombre");
                String apellidos = rs.getString("apellidos");
                int goles = rs.getInt("goles");
                System.out.println(id + " " + nombre + " " + apellidos + " " + goles);
            }
        } catch (SQLException ex) {
            System.out.println("Conexión fallida");
        }
    }
}
```

```java
public class Departamentos  implements java.io.Serializable {

     private int deptNo;
     private String dnombre;
     private String loc;
     private Set empleadoses = new HashSet(0);

    public Departamentos() {}
    public Departamentos(int deptNo) {}
    public Departamentos(int deptNo, String dnombre, String loc, Set empleadoses) {}

    public int getDeptNo() {}
    public void setDeptNo(int deptNo) {}

    public String getDnombre() {}
    public void setDnombre(String dnombre) {}

    public String getLoc() {}
    public void setLoc(String loc) {}

    public Set getEmpleadoses(){}
    public void setEmpleadoses(Set empleadoses) {}
}
```

---

```java
public class Empleados implements java.io.Serializable {

    private int empNo;
    private Departamentos departamentos;
    private String apellido;
    private String oficio;
    private Integer dir;
    private Date fechaAlt;
    private Double salario;
    private Double comision;
    private int deptNo;

    public Empleados() {}
    public Empleados(int empNo, int deptNo, Departamentos departamentos) {}
    public Empleados(int empNo, String apellido, String oficio, Integer dir, Date fechaAlt, Double salario, Double comision, int deptNo) {}

    public int getEmpNo() {}
    public void setEmpNo(int empNo) {}

    public String getApellido() {}
    public void setApellido(String apellido) {

    public String getOficio() {}
    public void setOficio(String oficio) {

    public Integer getDir() {}
    public void setDir(Integer dir) {}

    public Date getFechaAlt() {}
    public void setFechaAlt(Date fechaAlt) {}

    public Double getSalario() {}
    public void setSalario(Double salario) {}

    public Double getComision() {}
    public void setComision(Double comision) {}

    public int getDeptNo() {}
    public void setDeptNo(int deptNo) {}

    public Departamentos getDepartamentos() {}
    public void setDepartamentos(Departamentos departamentos) {}
}
```

```java
public class ORM_Main {

    public static void main(String[] args) {

        // En esta clase principal solo vamos a insertar objetos

        //Obtener la sesión de Hibernate
        SessionFactory sesion = HibernateUtil.getSessionFactory();
        //Crear la sesión actual
        Session session = sesion.openSession();
        //Crear la transacción de la sesión
        Transaction tx = session.beginTransaction();

        System.out.println("Inserto una fila en la tabla DEPARTAMENTOS");
        // Crear un objeto departamento
        Departamentos dep = new Departamentos();
        // Le damos valores a sus atributos
        dep.setDeptNo((byte) 62);
        dep.setDnombre("MARKETING");
        dep.setLoc("GUADALAJARA");

        System.out.println("Insertar un EMPLEADO en el DEPARTAMENTO");
        // Crear un objeto de empleados
        Empleados em = new Empleados();
        // Le damos valores a sus atributos
        em.setEmpNo((short) 6677);
        em.setApellido("Kenva Walker");
        em.setDir(7499);
        em.setOficio("NBA player");
        em.setSalario((double) 2000);
        em.setComision((double) 10);

        // Se crea un objeto Departamentos para asignárselo al empleado
        Departamentos depEm = new Departamentos();
        depEm.setDeptNo((byte) 10);
        em.setDepartamentos(depEm);

        // Anidamos dos bloques try-catch para gestionar las posibles excepciones
        try {
            // Guardar el objeto
            session.save(em);
            try {
                // Commit de la transacción actual. Necesario para almacenar los datos en la BD
                tx.commit();
                // Si ejecutamos la transacción más de una vez,
                // se produce un error porque el objeto ya existe
            } catch (ConstraintViolationException e) {
                System.out.println("Empleado duplicado");
                System.out.println("Error SQL: " + e.getSQL());
            }
            // Si el objeto que estamos guardando no existe
        } catch (TransientPropertyValueException e) {
            System.out.println("Departamento no existe");
            System.out.println("Mensaje: " + e.getMessage());
        }
        // Cerrar la sesión
        session.close();
        System.exit(0);
    }
}
```

```java
public class Ejer_list_iterate {

    public static void main(String[] args) {
        // En esta clase vamos a hacer consultas con list() e iterate()
        // que devuelvan todos los registros de cada tabla

        SessionFactory sesion = HibernateUtil.getSessionFactory();
        Session session = sesion.openSession();
        // No necesitamos un Transaction porque no vamos a introducir datos

        // CONSULTA Departamentos con .list()
        // Creamos la consulta queryDep en la sesión, con una sentencia HQL
        Query queryDep = session.createQuery("from Departamentos");
        // El método list() devuelve en una colección todos los resultados de la consulta
        List<Departamentos> lista = queryDep.list();

        // Creamos un iterador a partir de la lista
        // El objeto iterador permite recorrer secuencialmente los elementos de la colección
        Iterator<Departamentos> iterDep = lista.listIterator();
        // Mostrar el tamaño de la lista
        System.out.println("Número de departamentos: " + lista.size());

        // Recorremos la lista hasta que se acaben los registros
        while (iterDep.hasNext()) {
            // Extraemos el objeto
            Departamentos dep = iterDep.next();
            // y mostramos por consola algunos de sus atributos
            System.out.println(dep.getDeptNo() + " " + dep.getDnombre() + " " + dep.getLoc());
        }

        //------------------------------------------------------------------------

        // CONSULTA Empleados con .iterate()
        // Con este método no podemos saber el tamaño de la lista
        Query qEmp = session.createQuery("from Empleados");
        qEmp.setFetchSize(10); //No hace nada
        Iterator iterEmp = qEmp.iterate();
        while (iterEmp.hasNext()) {
            // Casteo obligatorio porque iterEmp no sabe a qué objeto nos estamos refiriendo
            Empleados emp = (Empleados) iterEmp.next();
            System.out.println(emp.getApellido() + " " + emp.getSalario() + " " + emp.getOficio());
        }

        session.close();
        System.exit(0);
    }
}
```

```java
public class Ejer_Dep20 {

    public static void main(String[] args) {
        // Realiza una consulta que visualice el apellido y salario de los
        // empleados del departamento 20

        SessionFactory sesion = HibernateUtil.getSessionFactory();
        Session session = sesion.openSession();

        // Opción "A LO BESTIA"
        Query query = session.createQuery("from Empleados");
        List<Empleados> listaEmpleados = query.list();
        Iterator<Empleados> iterador = listaEmpleados.iterator();

        while (iterador.hasNext()) {
            Empleados emp = iterador.next();
            /* Hasta aquí, todo igual que en Ejer_list_iterate.java. Disponemos
             de todos los resultados de la consulta a la tabla Empleados.
             Ahora, hay que filtrar por numero de departamento
             Sacamos un objeto de tipo departamentos y lo asignamos al objeto
             Empleados a través del método .getDepartamentos()
             Este atributo "Departamentos" en Empleados establece la relación
             entre las dos tablas
             */
            Departamentos dep = emp.getDepartamentos();
            // Si el numero de departamento es igual a 20...
            if (dep.getDeptNo() == 20) {
                // mostramos el apellido y salario de los empleados
                System.out.println(emp.getApellido() + " " + emp.getSalario());
            }
            // Si además queremos mostrar los empleados de una localización:
            if (dep.getLoc().equalsIgnoreCase("BARCELONA")) {
                System.out.println(emp.getApellido() + " " + emp.getSalario());
            }
        }
        //-------------------------------------------------------------------------

        // Opción "OBJETOS EN SENTENCIA HQL"
        Query query2 = session.createQuery("from Empleados as e where "
                + "e.departamentos.loc='BARCELONA'");

        listaEmpleados = query2.list();

        iterador = listaEmpleados.iterator();

        while (iterador.hasNext()) {
            Empleados emp = iterador.next();
            System.out.println(emp.getApellido() + " " + emp.getSalario());
        }

        session.close();
        System.exit(0);
    }
}
```

```java
public class Ejer_ConsultasParametizadas {

    public static void main(String[] args) {

        // Realiza una consulta que utilice el parámetro :numeroEmpleado y muestre
        // el apellido y oficio del empleado que busque el usuario

        // Abrimos la sesión
        SessionFactory sesion = HibernateUtil.getSessionFactory();
        Session session = sesion.openSession();

        // Creamos la consulta en una sentencia de HQL. La variable se pone con ":" delante
        Query query = session.createQuery("from Empleados where empNo = :numeroEmpleado");

        System.out.println("Introduce el número de empleado que quieres ver: ");
        // Abrimos teclado
        Scanner teclado = new Scanner(System.in);
        // Recogemos el dato dado por el usuario
        int numEmp = teclado.nextInt();

        // Añadimos el parámetro a la consulta con un .setParameter(), este método
        // inserta el valor del dato dado por el usuario en el campo declarado en la sentencia
        query.setParameter("numeroEmpleado", numEmp);
        // Se hace un casteo del objeto porque no se ha definido el tipo de lista anteriormente
        // Si sabemos que la consulta nos va a devolver un único resultado, usamos .uniqueResult()
        Empleados emp = (Empleados) query.uniqueResult();

        // Obtenemos y mostramos apellido y oficio del empleado
        System.out.println(emp.getApellido() + " " + emp.getOficio());

        //------------------------------------------------------------------------

        // Consulta los empleados cuyo número de departamentos es 30 y el oficio VENDEDOR.

        // Definimos la sentencia HQL fuera de la query porque nos piden más de un parámetro
        String hqlStatement = "from Empleados emp where emp.departamentos.deptNo = :numDep "
                + "and emp.oficio = :ofi";

        // Metemos la sentencia en la query. Más limpio
        Query query2 = session.createQuery(hqlStatement);

        // Añadimos los parámetros a la query relacionando los valores con los campos declarados
        query2.setParameter("numDep", 30);
        query2.setParameter("ofi", "VENDEDOR");

        // Al ser más de uno los resultados, se meten en una lista y se crea el iterador
        List<Empleados> lista = query2.list();
        Iterator<Empleados> iterador = lista.iterator();

        // Se recorren los resultados con el iterador y se muestran
        while (iterador.hasNext()) {
            Empleados emp2 = (Empleados) iterador.next();
            System.out.println(emp2.getApellido() + " " + emp2.getOficio());
        }

        // Cerramos la sesión
        session.close();
        System.exit(0);
    }
}
```

```java
public class Ejer_ClasesNoAsociadas {

    public static void main(String[] args) {

        // Realiza una consulta que relacione los apellidos de los empleados
        // con la localización de los departamentos a los que pertenecen

        SessionFactory sesion = HibernateUtil.getSessionFactory();
        Session session = sesion.openSession();

        // Esta sentencia une las dos tablas mediante el atributo departamentos de la tabla Empleados
        String hqlStatement = "from Empleados e, Departamentos d where "
                + "e.departamentos.deptNo = d.deptNo order by Apellido";

        Query query = session.createQuery(hqlStatement);
        Iterator iterador = query.iterate();

        while (iterador.hasNext()) {
            // En cada iteración se obtiene un array de objetos que representa una fila de resultados
            Object[] par = (Object[]) iterador.next();
            // El primer elemento del array (par[0]) hace referencia a un objeto de tipo Empleados
            Empleados emp = (Empleados) par[0];
            // El segundo elemento del array (par[1]) se refiere a un objeto de tipo Departamentos
            Departamentos dep = (Departamentos) par[1];
            // Mostramos el apellido del empleado con la localización del departamento
            System.out.println(emp.getApellido() + ", " + dep.getLoc());
        }

        session.close();
        System.exit(0);
    }
}
```

```java
public class Ejer_ClaseUnicoResultado {

    public static void main(String[] args) {

        // Realiza una consulta que muestre la media del slario de los empleados

        SessionFactory sesion = HibernateUtil.getSessionFactory();
        Session session = sesion.openSession();

        // Se usa la función avg() dentro de la sentencia
        String hqlStatement = "select avg(e.salario) from Empleados as e";
        Query query = session.createQuery(hqlStatement);

        // No es necesario iterar ni recorrer resultados porque sabemos que solo hay uno
        // Asignamos el resultado de la consulta a la variable media. Obligatorio castear
        Double media = (Double) query.uniqueResult();
        System.out.println("La media de salarios es: " + media);

        session.close();
        System.exit(0);
    }
}
```

```java
public class Ejer_ClaseNoAsociada_ResultadoMultiple {

    public static void main(String[] args) {

        SessionFactory sesion = HibernateUtil.getSessionFactory();
        Session session = sesion.openSession();

        String hqlStatement = "select e.departamentos.deptNo, "
                + "avg(e.slario), count(e.pellido), sum(e.salario) "
                + "from Empleados as e group by e.departamentos.deptNo";

        Query query = session.createQuery(hqlStatement);
        Iterator iterador = query.iterate();

        while (iterador.hasNext()) {
            Object[] par = (Object[]) iterador.next();
            Double mediaSalario = (Double) par[0];
            Long numEmpleados = (Long) par[1];
            Double sumaSalario = (Double) par[2];
            System.out.println(mediaSalario + " " + numEmpleados + " " + sumaSalario);
        }

        session.close();
        System.exit(0);

    }
}
```

```java
public class Ejer_Update_Delete {

    public static void main(String[] args) {

        SessionFactory sesion = HibernateUtil.getSessionFactory();
        Session session = sesion.openSession();
        // Tenemos que abrir y cerrar la transacción porque actualizamos el contenido
        Transaction tx = session.beginTransaction();

        // UPDATE
        String hqlUpdate = "update Empleados set salario = 500 where emp_no = 6677";
        Query query = session.createQuery(hqlUpdate);

        int filas = query.executeUpdate();

        System.out.println("Se han modificado todas estas filas: " + filas);

        // DELETE
        String hqlDelete = "delete Empleados where emp_no = 6677";
        Query query2 = session.createQuery(hqlDelete);

        int filas2 = query2.executeUpdate();
        System.out.println("Me he cargado " + filas2 + " empleados");

        tx.commit();
        session.close();
        System.exit(0);
    }
}
```