Tarea para AD04.

Enunciado.

Una empresa dispone de una base de datos contiene las tablas con la información necesaria para su gestión. Las tablas son las siguientes:

- ✓ Tabla DEPT que contiene información de los diferentes departamentos que tiene la empresa. La clave principal DEPTNO.
- ✓ Tabla EMP que contiene la información de los diferentes empleados que tiene la empresa. Tiene como clave principal EMPNO y ajenas DEPTNO que relaciona con la tabla departamentos y MGR que establece la relación con la misma tabla mostrando ser jefe de.

La creación de las tablas y sus relaciones se dan en el fichero **tablas.sql** que se adjunta.

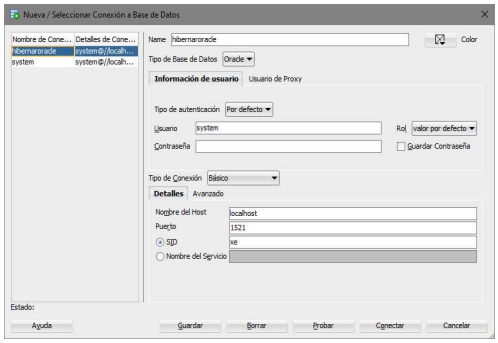
La base de datos que se utilizará será Oracle.

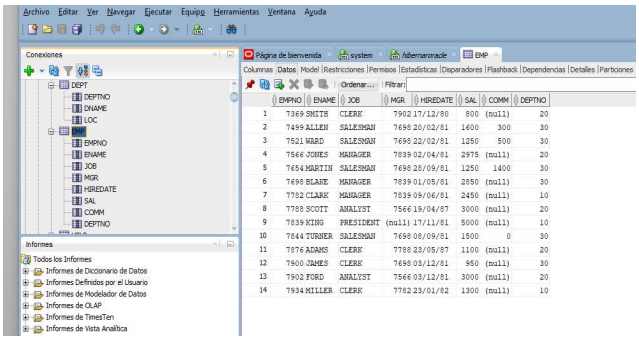
Mapea las tablas utilizando Hibernate con NetBeans y realiza un proyecto Java llamado HibernateOracle que obtenga lo siguiente:

1. Crea la base de datos

Accedo a `SQL Developer` y desde la ventana de Conexiones, creamos la **nueva conexión**, que yo he llamado `hibernateoracle`.

Con ella creada, vamos a `Tablas` y desde su menú contextual, elegimos `importar` para que de forma automática nos cree las tablas, tras pasarle el fichero facilitado con la tarea.

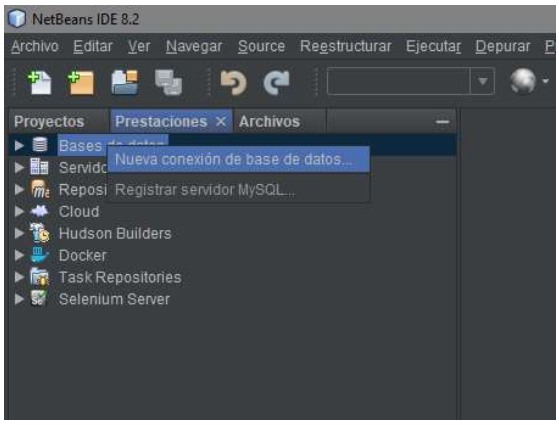


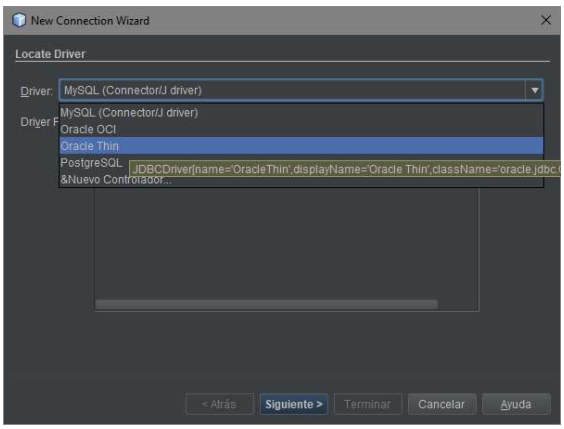


Con la base de datos generada, vamos a realizar las configuraciones necesarias para que nuestro proyecto pueda conectar con ella.

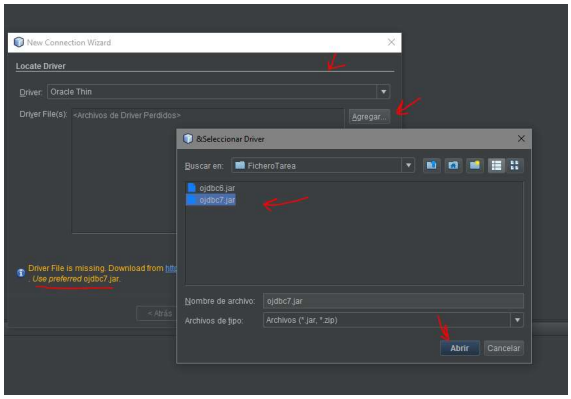
Abrimos **NetBeans** y desde `Prestaciones`, pulsamos para crear una `Nueva conexión de base de datos.`

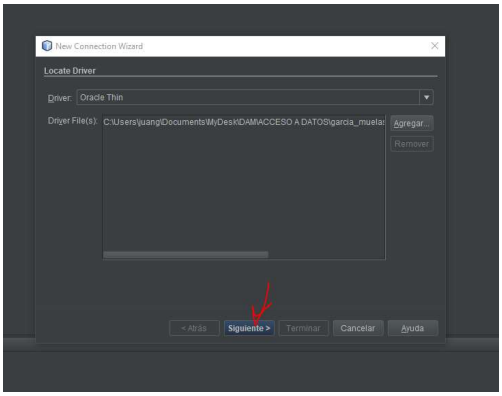
Desde ahí nos guía para elegir el **driver de conexión**, donde seleccionamos el de **Oracle**.



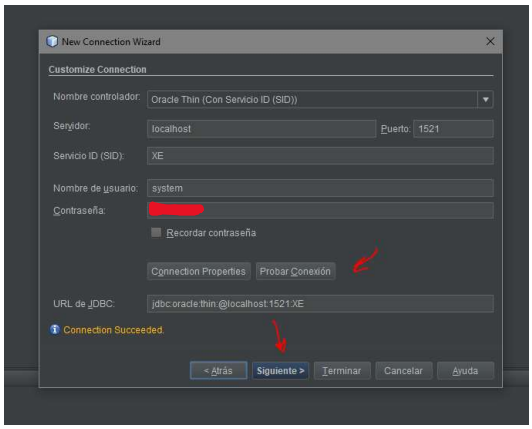


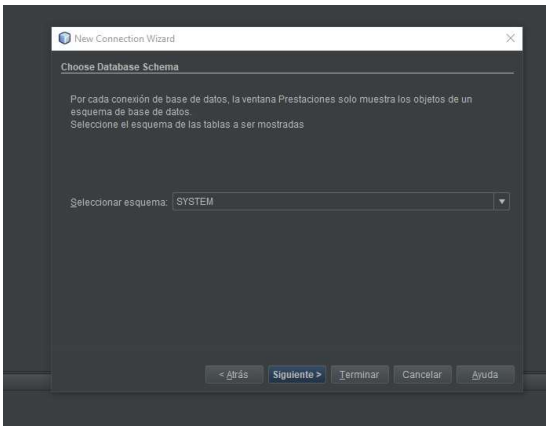
NetBeans nos recomienda para evitar errores no usar el que nos facilita la tarea, así que tras descargar el recomendado, lo aceptamos y seguimos hasta añadir nuestros datos de conexión.





Elegimos el controlador de **Oracle**, añadimos nuestro `usuario` y `password` para **Oracle** y tras probar la conexión, terminamos la configuración.

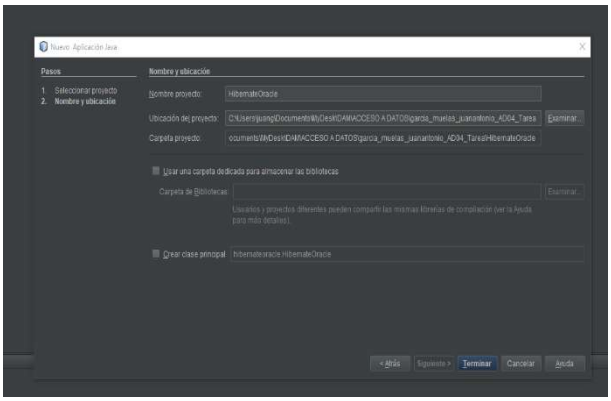




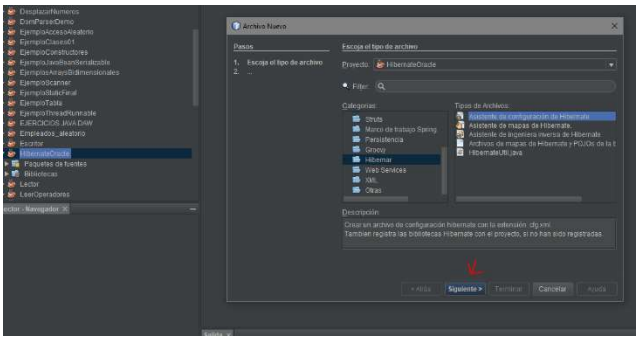
2. Configura y crea la ORM Hibernate.

Tras terminar con la conexión, creo el proyecto, llamado como piden en el enunciado `HibernateOracle`.

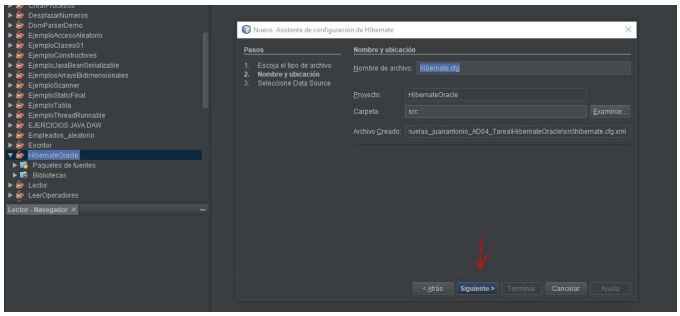
Al no venir muy claro desde el temario, lo creo sin clase principal, para así añadirla donde pueda resultar finalmente oportuna.



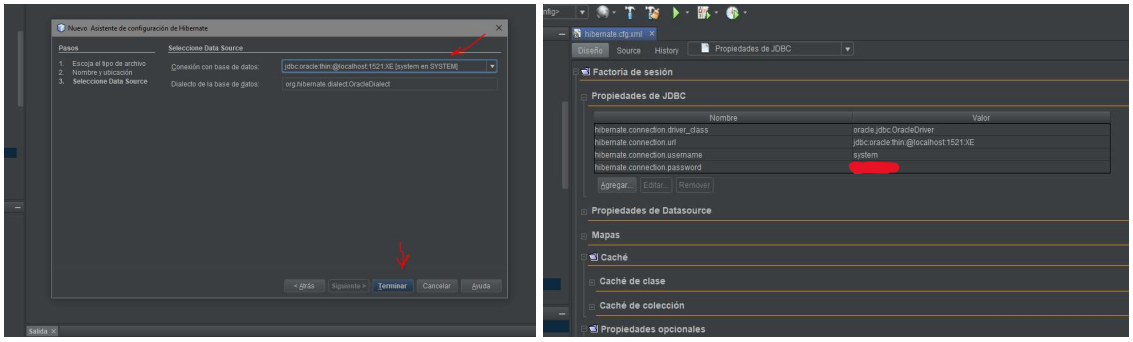
NetBeans 8.2, como bien apuntan desde el temario, ya incorpora `Hibernate`, por lo que en ese aspecto, no tendremos que seguir ningún tipo de tutorial de instalación. Vamos al proyecto y creamos el **archivo de configuración**, mediante `Hibernar>Asistente de configuración de Hibernate`.



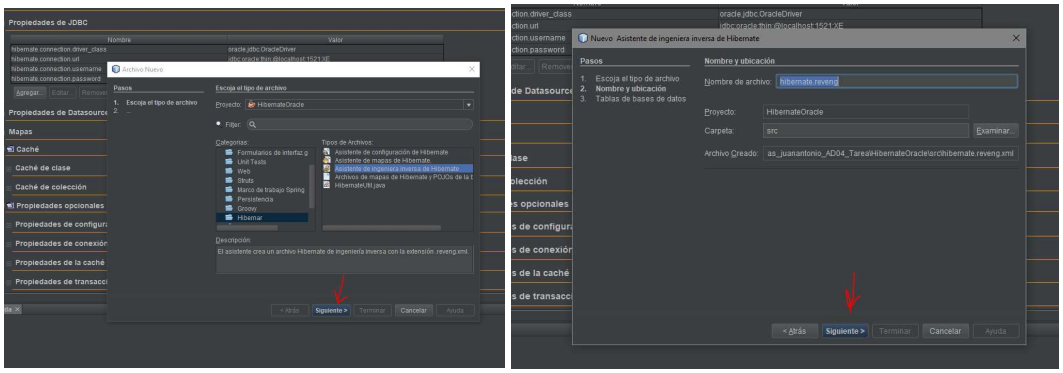
Le dejamos su nombre por defecto:



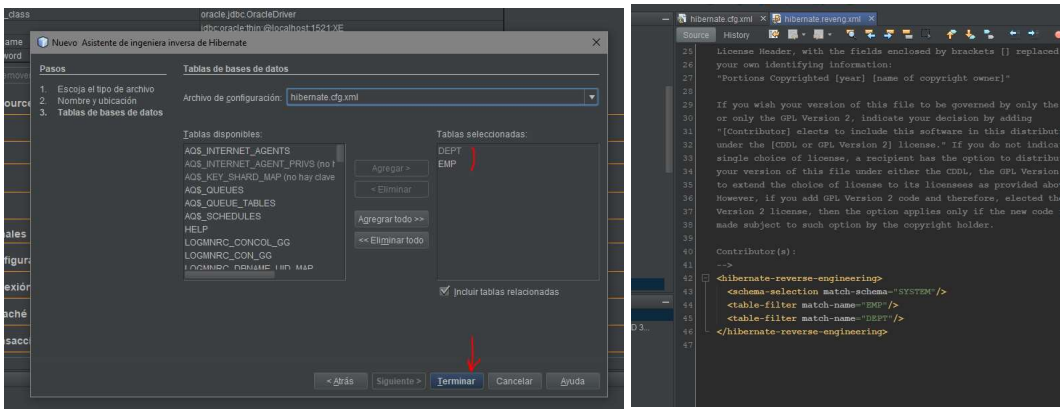
Este archivo, entre otras propiedades, incluye información sobre nuestra conexión a la base de datos. Por ello, elegimos la conexión creada en el paso anterior y nos genera una nueva clase con los datos:



El siguiente paso, es crear mediante Hibernar>Asistente de ingeniería inversa de Hibernate, el siguiente archivo de configuración para las **relaciones entre tablas**:

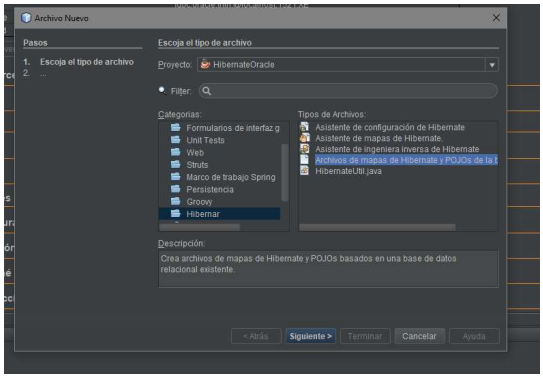
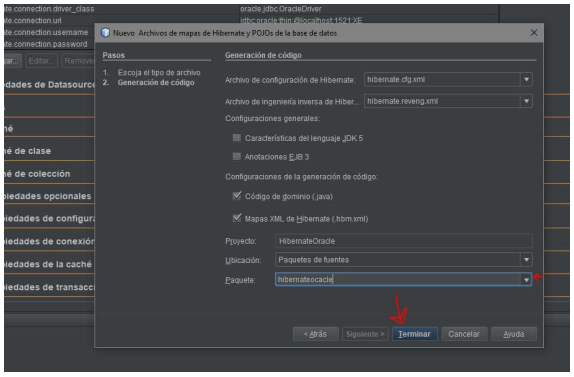


Seleccionamos las tablas para la tarea y nos genera nuestro xml.

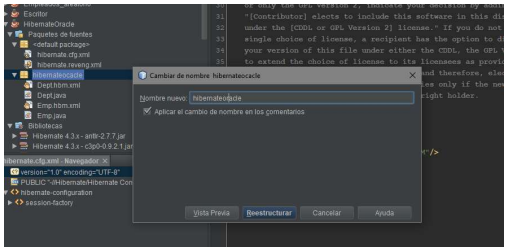


Siguiendo el orden creado por el mismo **Netbeans**, generamos los `Archivos de mapas de Hibernate y POJOs` basados en una base de datos relacional existente, es decir, **generamos las clases java**, correspondientes a las tablas de la base de datos y que serán las que usaremos para comunicarnos.

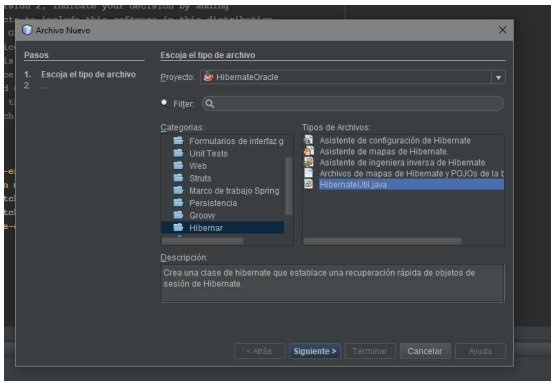
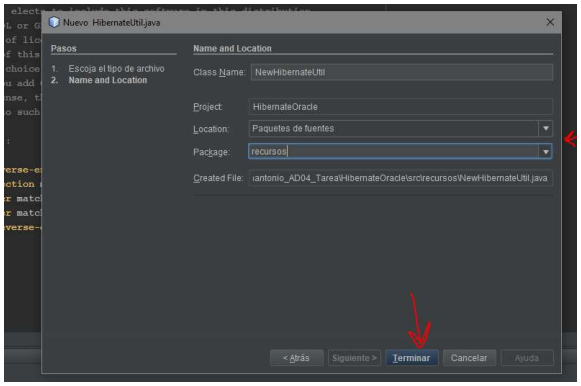
Pide un paquete para añadirlo, así que se crea uno.

Veo que al crear el nuevo paquete para los archivos, he puesto mal ortográficamente el nombre, así que lo modifico por legibilidad y para evitar posibles errores al codificar luego.

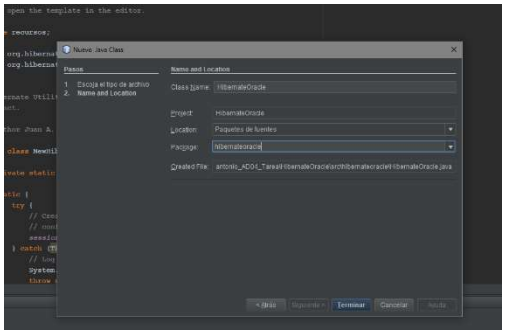



Por último, creamos nuestro archivo `HibernateUtil` para manejar nuestra **conexión**, que podemos añadirlo en la carpeta que queramos, pero por si crece más el proyecto, lo añado a un nuevo paquete que llamo `recursos`.

3. Realiza una inserción y un borrado sobre la tabla EMP.

Tras la configuración, creamos una **nueva clase** llamada `HibernateOracle` para manejar las peticiones, intentando seguir para ello las pautas del pdf de ayuda facilitado desde el temario.



Esta clase, para no extender más el ejercicio y simplificar su explicación, va a contener los **métodos de acceso** a los objetos creados, así como los métodos para **añadir o borrar** registros de nuestra base de datos.

Como en ejercicios anteriores, he intentado que quede suficientemente documentado el código de la misma para su mejor comprensión.

```
/*
 * TAREA AD04.
 * Una empresa dispone de una base de datos contiene las tablas con
 * la información necesaria para su gestión.
 * Las tablas son las siguientes:
 * ⎩ Tabla DEPT que contiene información de cada departamento
 *   que tiene la empresa. La clave principal DEPTNO.
 * ⎩ Tabla EMP que contiene la información de los diferentes empleados
 *   que tiene la empresa. Tiene como clave principal EMPNO y ajenas
 *   DEPTNO que relaciona con la tabla departamentos y MGR que
 *   establece la relación con la misma tabla mostrando ser jefe de.
 *
 * La base de datos que se utilizará será Oracle.
 *
 * Mapea las tablas utilizando Hibernate con NetBeans y realiza un
 * proyecto Java llamado HibernateOracle que obtenga lo siguiente:
 * 1. Crea la base de datos.
 * 2. Configura y crea la ORM Hibernate.
 * 3. Realiza una inserción y un borrado sobre la tabla EMP.
 * 4. Obtener un listado sobre las tablas EMP y DEPT que visualice
 *    empno, ename, sal, dname y loc.
 */
package hibernateoracle;

import java.math.BigDecimal;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import org.hibernate.Query;
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import recursos.NewHibernateUtil;

/**
 *
 * @author Juan A. García Muelas <juangmuelas@gmail.com>
 * @version 1
 * @since 12/01/22
 */
public class HibernateOracle {
    /**
     * Constructor vacío
     */
    public HibernateOracle() {
    }

    /**
     * Antes de crear el main, implementamos la instancia para
     * trabajar con la BD, mediante el patrón Singleton
     */
    private Session sesion;
    SessionFactory sesionFact;
```

```
    //Obtenemos la sesión asegurando iniciarla si es null
    public Session getSession() {
        if (sesion == null) {
            SessionFactory sessionFactory =
            NewHibernateUtil.getSessionFactory();
            sesion = sessionFactory.openSession();
        }
        return sesion;
    }

    /**
     * Creamos un método genérico que nos guarda en una lista los
     * objetos query devueltos y así hacer más legible la
     * creción y lectura de código en cada petición.
     * @param ConsultaSQL String que recoge la query.
     */

    public <T> List<T> consulta(String ConsultaSQL) {
        Query query = getSession().createQuery(ConsultaSQL);
        return query.list();
    }
```

Los métodos para **añadir** (`insertarEmpl()`) y para **borrar** (`borrarEmpl()`), mantienen el patrón facilitado desde el tema.

```
    public void insertarEmpl(Emp empleado) {
        /**
         * PASOS:
         * Para utilizar la persistencia en Hibernate, SessionFactory
         * Comenzamos luego la transacción
         * save para introducir el objeto
         * commit para realizar la transacción y sincronizar con la BD
         * Cierra sesión
         */

        SessionFactory sesion = NewHibernateUtil.getSessionFactory();
        Session session = sesion.openSession();
        Transaction transaccion = session.beginTransaction();
        session.save(empleado);
        transaccion.commit();
        session.close();
    }

    public void borrarEmpl(Emp empleado) {
        /**
         * PASOS:
         * Para utilizar la persistencia en Hibernate, SessionFactory
         * Comenzamos luego la transacción
         * delete para borrar objetos persistentes
         * commit para realizar la transacción y sincronizar con la BD
         * Cierra sesión
         */

        SessionFactory sesion = NewHibernateUtil.getSessionFactory();
        Session session = sesion.openSession();
        Transaction transaccion = session.beginTransaction();
        session.delete(empleado);
        transaccion.commit();
        session.close();
    }
```

En el método `main()`, creamos un objeto para mostrar el funcionamiento, respetando los tipos de datos y formato incluidos al crear los archivos de comunicación con nuestra base de datos.

Asímismo, dado que todos los ejemplos facilitados no muestran interactividad entre el usuario y el programa, y para no salirnos del objetivo de la tarea, he decidido facilitar los datos a través del propio código de ejemplo.

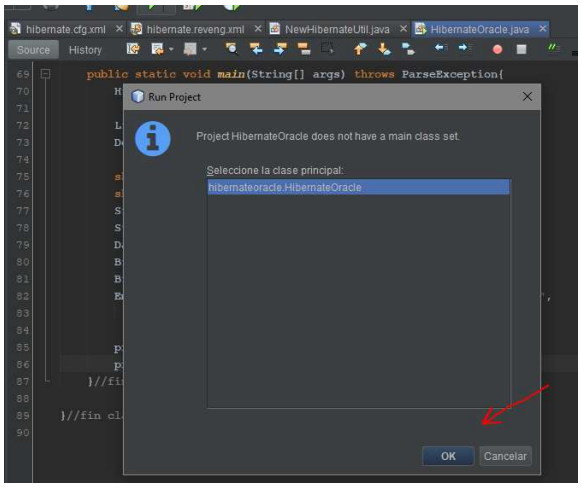
```
    public static void main(String[] args) throws ParseException{
        HibernateOracle pruebahibernate = new HibernateOracle();

        /**
         * Aprovechamos que Hibernate soporta NO incluir el SELECT
         * para acortar la consulta.
         * Obtenemos los departamentos.
         */
        List<Dept> deps = pruebahibernate.consulta("FROM Dept");
        Dept departament = deps.get(0);

        /**
         * Visto el tipo de datos que se utilizan para guardar
         * cada campo, genero las siguientes variables.
         * @param short numemp recoge el número de empleado
         * @param short numdep recoge el número de departamento
         * al que es asignado.
         * @param String nombre recoge nombre empleado (la BD
         * facilitada sólo ofrece un tamaño de 10 bytes).
         * @param String puesto recoge cargo o puesto del empleado (la
         * BD facilitada sólo ofrece un tamaño de 9 bytes).
         * @param String valorFecha recoge la fecha de entrada (atento
         * con el formato que nos permite: yyyy-mm-dd ).
         * @param fecha objeto Date que recoge el string anterior.
         * @param BigDecimal salario recoge el valor entero del sueldo
         * @param BigDecimal comision recoge el valor entero comisión.
         *
         */

        short numemp = 7999;
        short numdep = 7566;
        String nombre ="GARCIA";
        String puesto ="DEVELOPER";
        String valorFecha = "2022-01-12";
        //recordar el tipo de formato para fechas
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        Date fecha = sdf.parse(valorFecha);
        BigDecimal salario = BigDecimal.valueOf(1800);
        //por registrar un valor en comisión, pero puede ser cero
        BigDecimal comision = BigDecimal.valueOf(50);


        //creamos un nuevo objeto empleado.
        Emp datosEmpleado = new Emp(numemp, departament, nombre,
        puesto,numdep, fecha, salario, comision);


        //llamamos a los métodos para crear y borrar
        pruebahibernate.insertarEmpl(datosEmpleado);
        pruebahibernate.borrarEmpl(datosEmpleado);
```

4. Obtener un listado sobre las tablas EMP y DEPT que visualice empno, ename, sal, dname y loc.

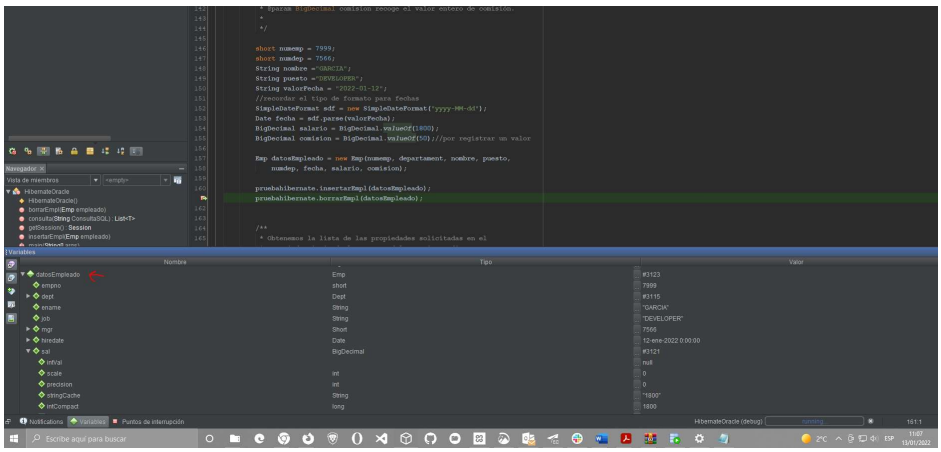
Con esta parte finalizamos la implementación del código necesario de la tarea, mediante un bucle for que recorre los objetos obtenimos de nuestra consulta HQL y los muestra por consola.

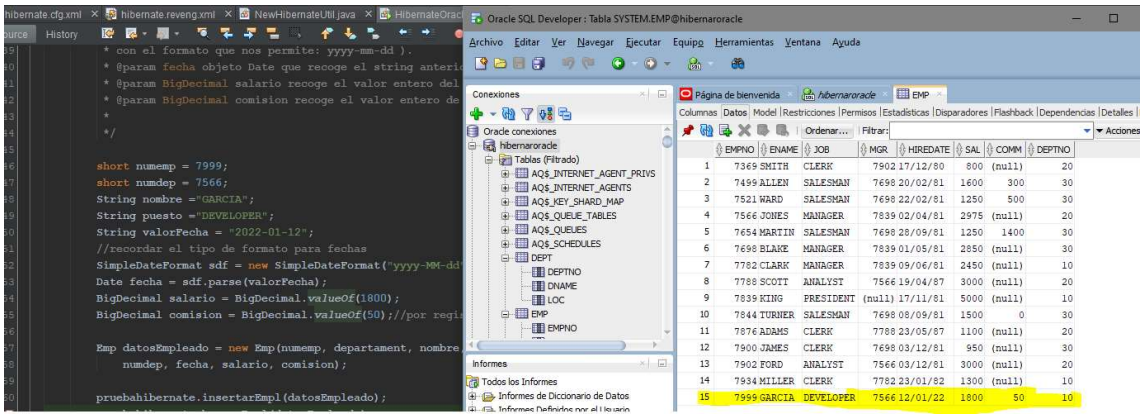
```
        /**
         * Obtenemos la lista de las propiedades solicitadas en el
         * enunciado siguiendo las pautas del temario, mediante una
         * Lista de Array de Objetos
         */
        List<Object[]> lista = pruebahibernate.consulta("FROM Emp as
            empleado INNER JOIN empleado.dept as departamento");
        for (Object[] objeto : lista) {
            Emp empleado = (Emp) objeto[0];
            Dept departamento = (Dept) objeto[1];
            System.out.println("NUMERO: " + empleado.getEmpno()
                    + "\t NOMBRE: " + empleado.getEname()
                    + "\t SUELDO: " + empleado.getSal()
                    + "\t LOCALIDAD: " + departamento.getLoc()
                    + "\t DEPARTAMENTO: " + departamento.getDname());
        }
    }//fin main

}//fin clase HibernateOracle
```

Para probar su funcionamiento, primero asignamos al proyecto esta clase como principal.



Si utilizamos puntos de interrupción, podemos comprobar como nos crea el nuevo objeto, y lo añade a la base de datos.





Tras dejarlo continuar, borra el registro y genera la lista solicitada