

**MP\_0486. Acceso a datos**

**UF6. Bases de datos orientadas a objetos**

El interfaz de  
programación de  
aplicaciones de la  
base de datos.  
Librería Neodatis.



# Índice

---

Objetivos .....	3
Crear BD Alumnos.....	4
Crear base de datos de alumnos en Java .....	4
Examinar base de datos desde NeoDatis .....	6
Distintas vistas para examinar la BD.....	8
Object View.....	9
Table View .....	10
Query .....	11
New Object .....	12
Consultar BD Alumnos .....	13
Consultar base de datos de alumnos en Java.....	13
Importar y exportar en Neodatis .....	15
Exportar Object DB a XML .....	15
Importar XML para obtener Object DB .....	17
Despedida .....	21
Resumen.....	21

# Objetivos

En esta lección perseguimos los siguientes objetivos:

- Desarrollar un programa Java capaz de crear una base de datos orientada a objetos, con ayuda de las clases de la librería de NeoDatis.
- Examinar los objetos almacenados en la base de datos desde la aplicación NeoDatis.
- Leer el contenido de una base de datos orientada a objetos desde Java, con ayuda de la librería NeoDatis.
- Exportar una base de datos orientada a objetos a formato XML.
- Importar un archivo XML para obtener una base de datos orientada a objetos.

# Crear BD Alumnos

## Crear base de datos de alumnos en Java

Ha llegado el momento de **crear la base de datos de alumnos con ayuda de la librería de clases de NeoDatis**, que importaste en la lección anterior.

Dentro del proyecto **ProyectoODBMS**, crea la clase **Principal**. Después, copia y pega el siguiente código:

```
import org.neodatis.odb.ODB;
import org.neodatis.odb.ODBuilderFactory;

public class Principal {

    public static void main(String[] args) {
        Alumno alu1 = new Alumno("Miguel", 25);
        Alumno alu2 = new Alumno("Pedro", 17);
        Alumno alu3 = new Alumno("Rosa", 19);

        alu1.calificar("Matemáticas", 45);
        alu1.calificar("Inglés", 70);
        alu1.calificar("TIC", 85);
        alu1.calificar("Física", 55);

        alu2.calificar("Matemáticas", 65);
        alu2.calificar("Inglés", 70);
        alu2.calificar("TIC", 95);
        alu2.calificar("Ciencias Naturales", 83);

        alu3.calificar("Matemáticas", 90);
        alu3.calificar("Inglés", 53);
        alu3.calificar("TIC", 75);
        alu3.calificar("Física", 25);

        ODB odbAlumnos = ODBuilderFactory.open("G:/ALUMNOS.DB");
        odbAlumnos.store(alu1);
        odbAlumnos.store(alu2);
        odbAlumnos.store(alu3);

        odbAlumnos.close();
        System.out.println("Se ha creado la DBOO con Ayuda de NeoDatis");
    }
}
```

**Vamos a analizar el código anterior paso a paso.**

### 1. En primer lugar, creamos tres objetos de la clase *Alumno*.

Creamos tres objetos de la clase *Alumno* con sus calificaciones asociadas. Cada alumno se examina de cuatro asignaturas.

```
Alumno alu1 = new Alumno("Miguel", 25);
Alumno alu2 = new Alumno("Pedro", 17);
Alumno alu3 = new Alumno("Rosa", 19);
```

```
alu1.calificar("Matemáticas", 45);  
alu1.calificar("Inglés", 70);  
alu1.calificar("TIC", 85);  
alu1.calificar("Física", 55);  
  
alu2.calificar("Matemáticas", 65);  
alu2.calificar("Inglés", 70);  
alu2.calificar("TIC", 95);  
alu2.calificar("Ciencias Naturales", 83);  
  
alu3.calificar("Matemáticas", 90);  
alu3.calificar("Inglés", 53);  
alu3.calificar("TIC", 75);  
alu3.calificar("Física", 25);
```

## 2. En segundo lugar, obtenemos la conexión con la base de datos *Alumnos*.

```
ODB odbAlumnos = ODBFactory.open("G:/ALUMNOS.DB");
```

El método **open** de la clase *ODBFactory* abre la base de datos orientada a objetos especificada en el fichero y retorna un objeto ODB, que representa dicha base de datos. Si no existe el fichero de base de datos especificado, será creado.

En el ejemplo hemos utilizado la ruta G:/ para guardar el archivo ALUMNOS.DB. Debes sustituir dicha ruta por la que desees, dentro del sistema de archivos de tu equipo local.

Veamos más detenidamente las dos clases empleadas en la línea de código que nos ocupa:

- **ODBFactory**: esta clase es imprescindible, ya que actúa de puente entre la capa de persistencia, suministrada por la librería NeoDatis, y el resto de capas de nuestra aplicación Java. Además, nos permite abrir una base de datos orientada a objetos a través del método estático *open*, retornando un objeto de la clase *ODB*.
- **ODB**: representa una conexión con una base de datos orientada a objetos almacenada en un fichero. A través del objeto *ODB*, en nuestro ejemplo *odbAlumnos*, podemos realizar las cuatro clásicas operaciones CRUD (*Create, Read, Update and Delete*).

## 3. En tercer lugar, persistimos los tres objetos *Alumno* creados.

El método **store** de la clase *ODB* recibe como argumento un objeto, que será persistido en la base de datos. Corresponde con la C (*Create*) de las cuatro operaciones CRUD.

```
odbAlumnos.store(alu1);  
odbAlumnos.store(alu2);  
odbAlumnos.store(alu3);
```

Persistimos tres objetos en la base de datos *ALUMNOS.DB*, es decir, los guardamos.

#### 4. En cuarto lugar, cerramos la conexión con la base de datos.

Una vez realizadas las operaciones deseadas, habrá que cerrar la conexión con la base de datos orientada a objetos a través del método *close* de la clase *ODB*.

```
odbAlumnos.close();
```

Si ejecutas varias veces el programa, se guardarán varias veces los mismos objetos, creando duplicados. Ten en cuenta que el método *open* crea la base de datos si no existe, pero si existe, la abre permitiendo añadir nuevos objetos sin eliminar los existentes.

## Examinar base de datos desde NeoDatis

En este apartado, volverás a **abrir la interfaz de usuario de NeoDatis para examinar la base de datos** que acabas de crear.

### 1. Abre la aplicación NeoDatis.

Accede a la carpeta donde descomprimiste NeoDatis y haz clic sobre el archivo ***odb-explorer.bat*** para abrir el entorno de usuario de *NeoDatis*.

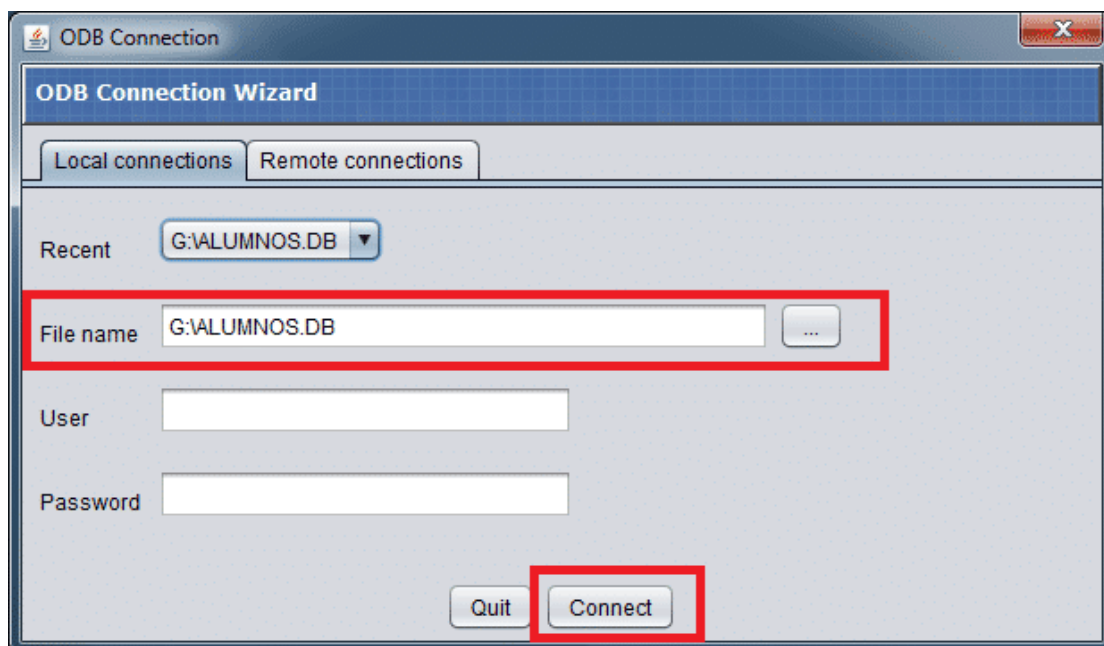
### 2. Abre la base de datos *ALUMNOS.DB*.

Para ello, selecciona en el menú **NeoDatis ODB / Open Database**.

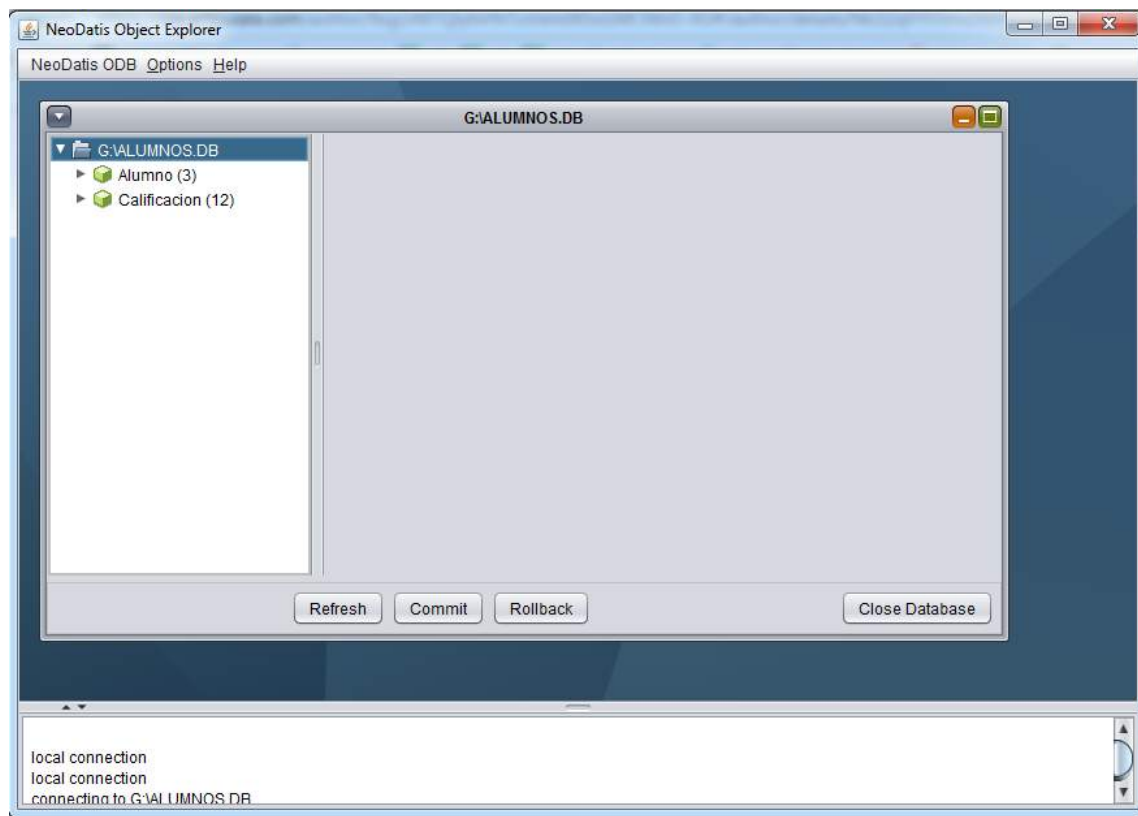




Debes especificar la ruta y el nombre de la base de datos en el campo **File Name**. Puedes hacer clic en el botón de los tres puntos, situado a la derecha, para obtener el cuadro de diálogo *Abrir* que te permitirá seleccionar el archivo.



Finalmente, haz clic en el botón **Connect** y tendrás una vista de la base de datos **ALUMNOS.DB**. Éste será el aspecto de la vista de la base de datos:



Aspecto final de tu base de datos.

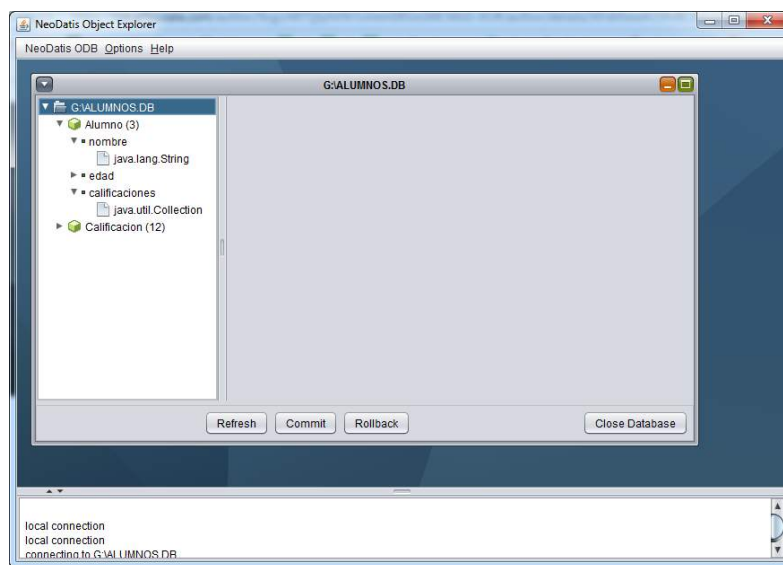
## Distintas vistas para examinar la BD

La interfaz de usuario de NeoDatis cuenta con **varias vistas para examinar los objetos almacenados en la base de datos.**

### ¿Quieres descubrirlos?

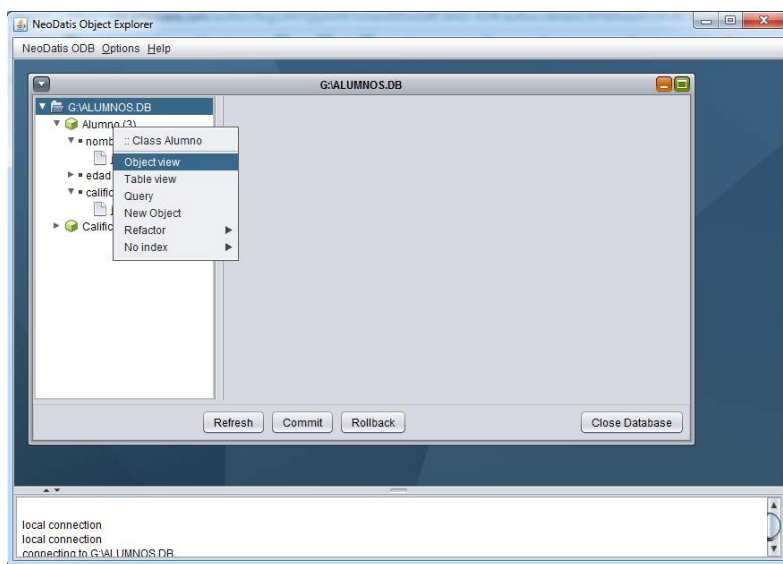
Lo que NeoDatis te está mostrando en un primer momento es la estructura de las clases a las que pertenecen los objetos almacenados en la base de datos.

Por la imagen, puedes comprobar que la base de datos tiene almacenados tres objetos *Alumno* y doce objetos *Calificacion*.



A la izquierda del nombre de la clase, hay un triángulo que te permitirá expandir o contraer los atributos que componen cada clase. Junto al atributo también aparece un triángulo, que te permitirá mostrar u ocultar el tipo de dato al que pertenece.

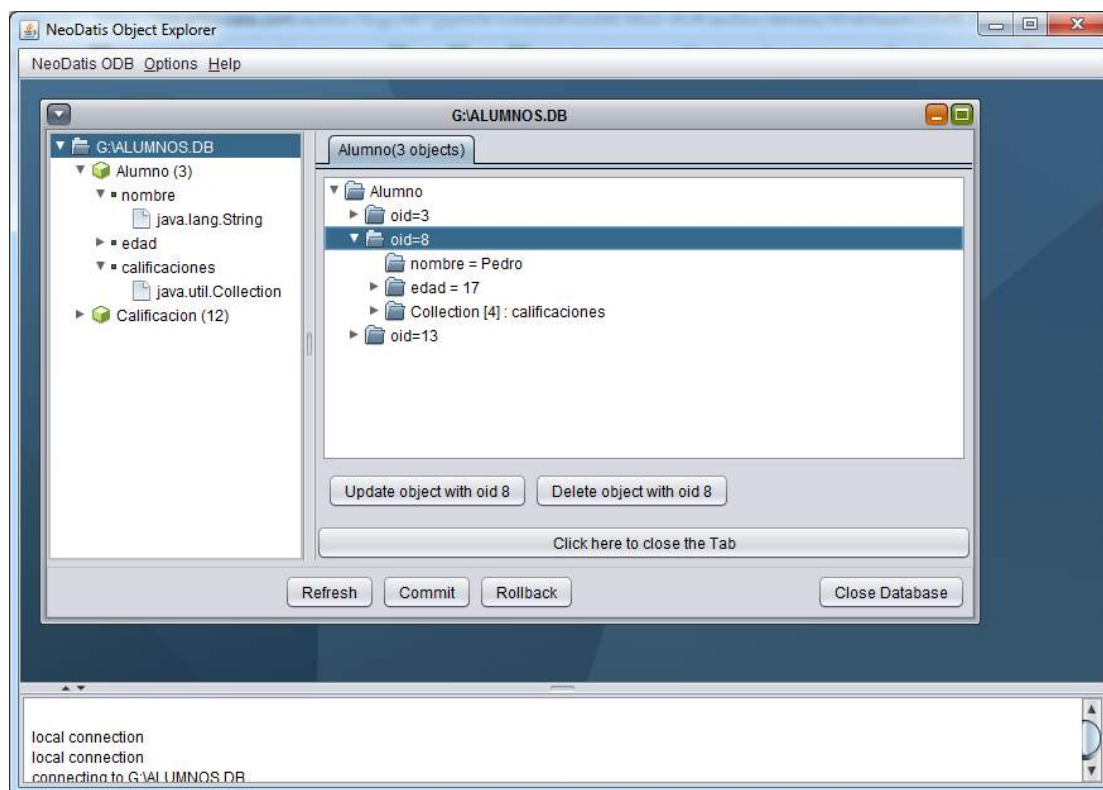
Puedes hacer clic derecho sobre el nombre de una de las clases para seleccionar una entre varias vistas.





## Object View

Selecciona la vista **Object View**.



*Object View.*

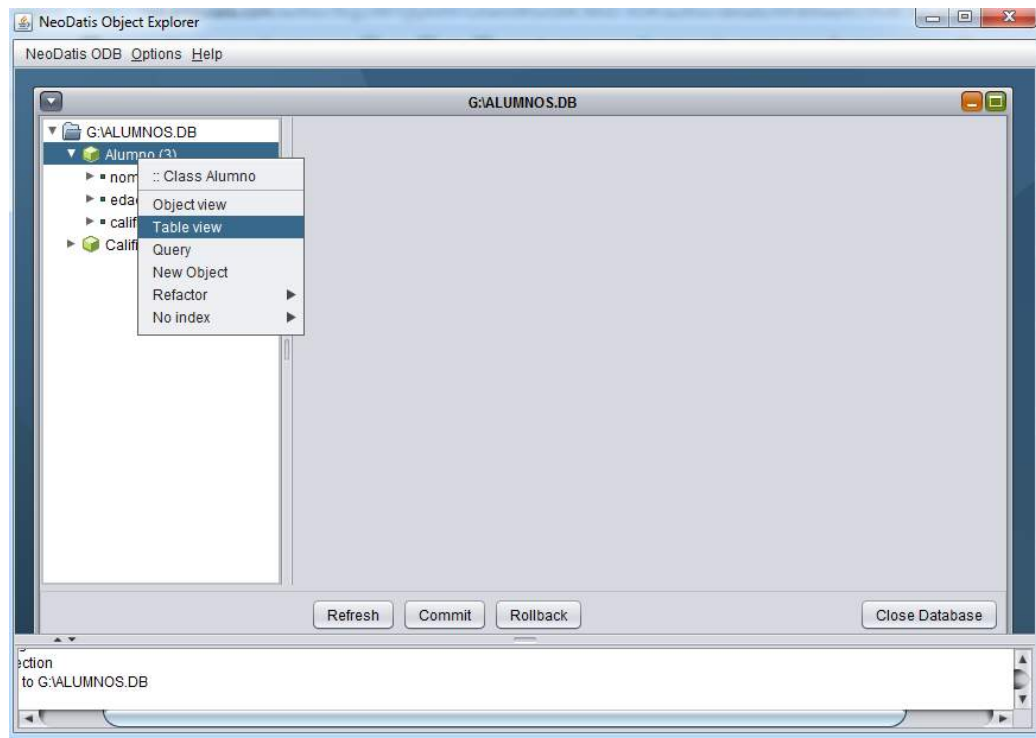
Puedes utilizar los botones *expandir* y *contraer* (representados por triángulos) para ver el objeto con más o menos detalle.

También puedes utilizar los botones *Update object* o *Delete object* para editar o eliminar el objeto seleccionado.

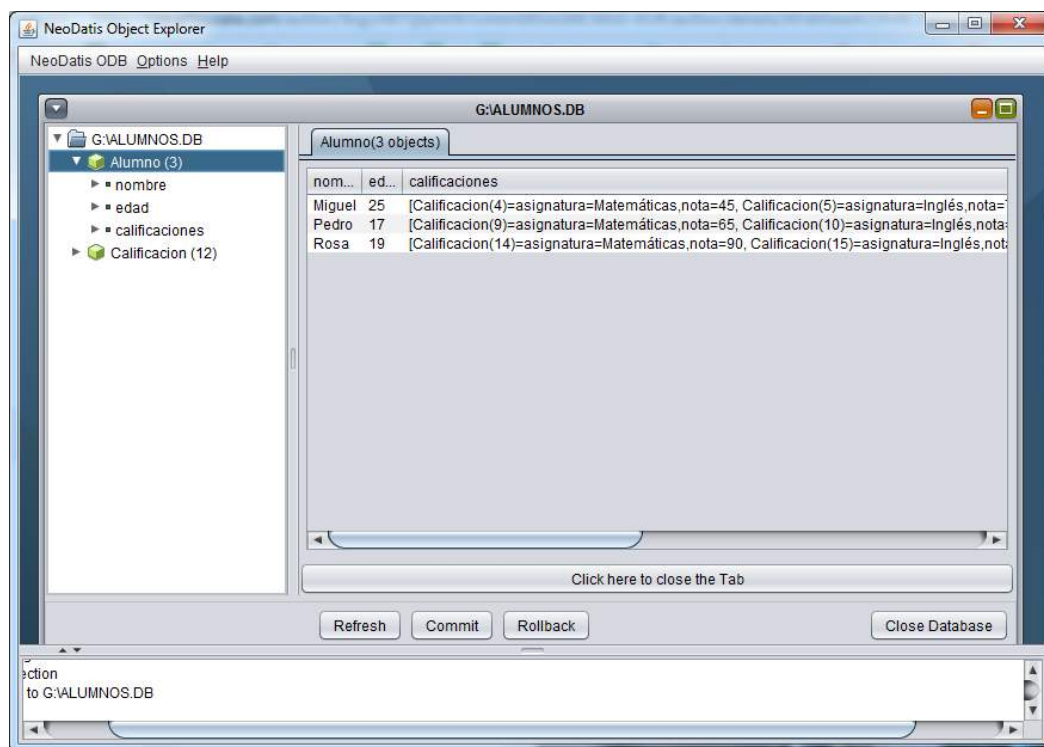
Cuando quieras cerrar la vista, sólo tienes que hacer clic en el botón *Click here to close the tab*.

## Table View

Vuelve a hacer clic derecho sobre la clase *Alumno*, pero esta vez selecciona la opción *Table View*.

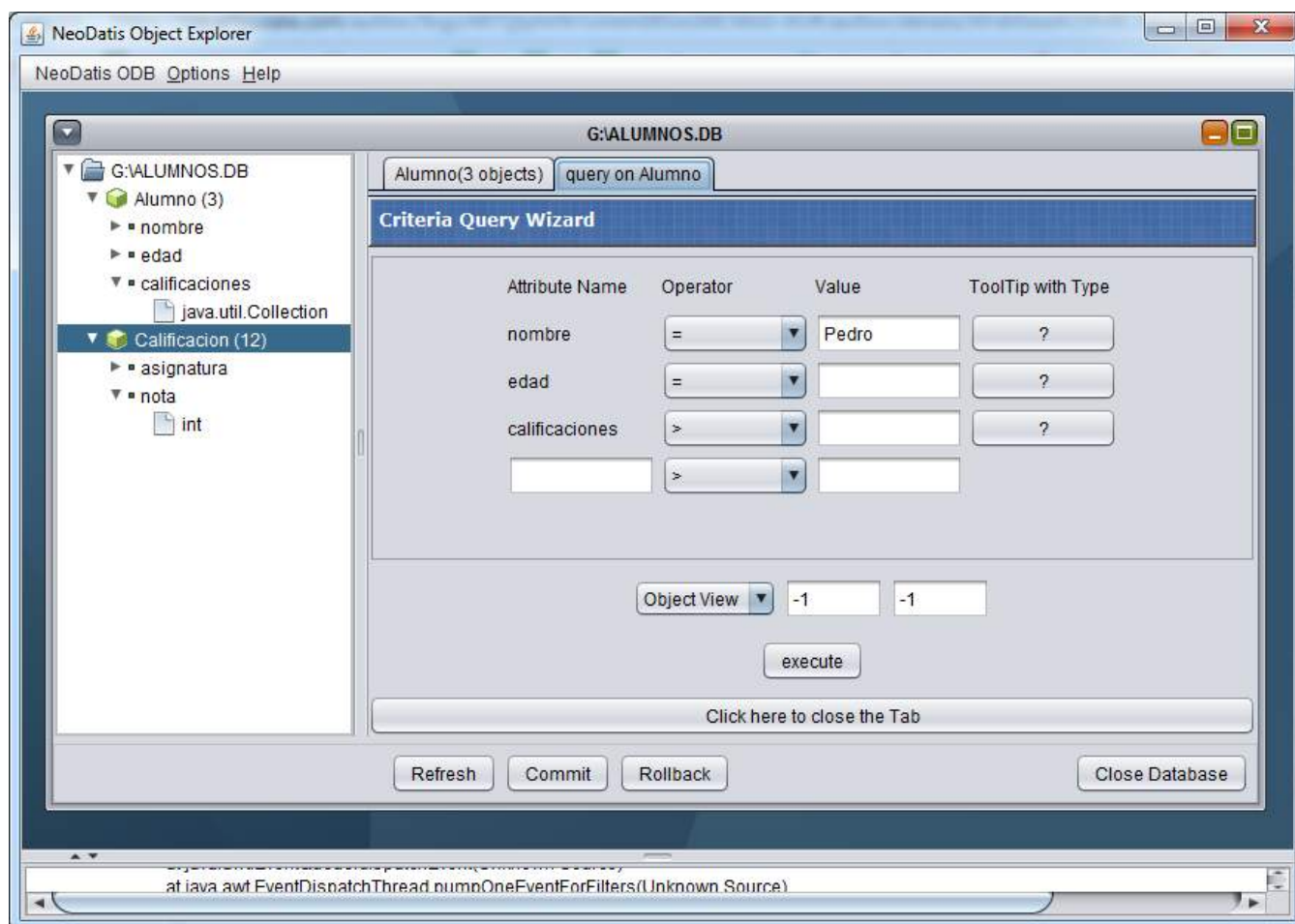


Ahora estás viendo los objetos organizados en una tabla con filas y columnas.



## Query

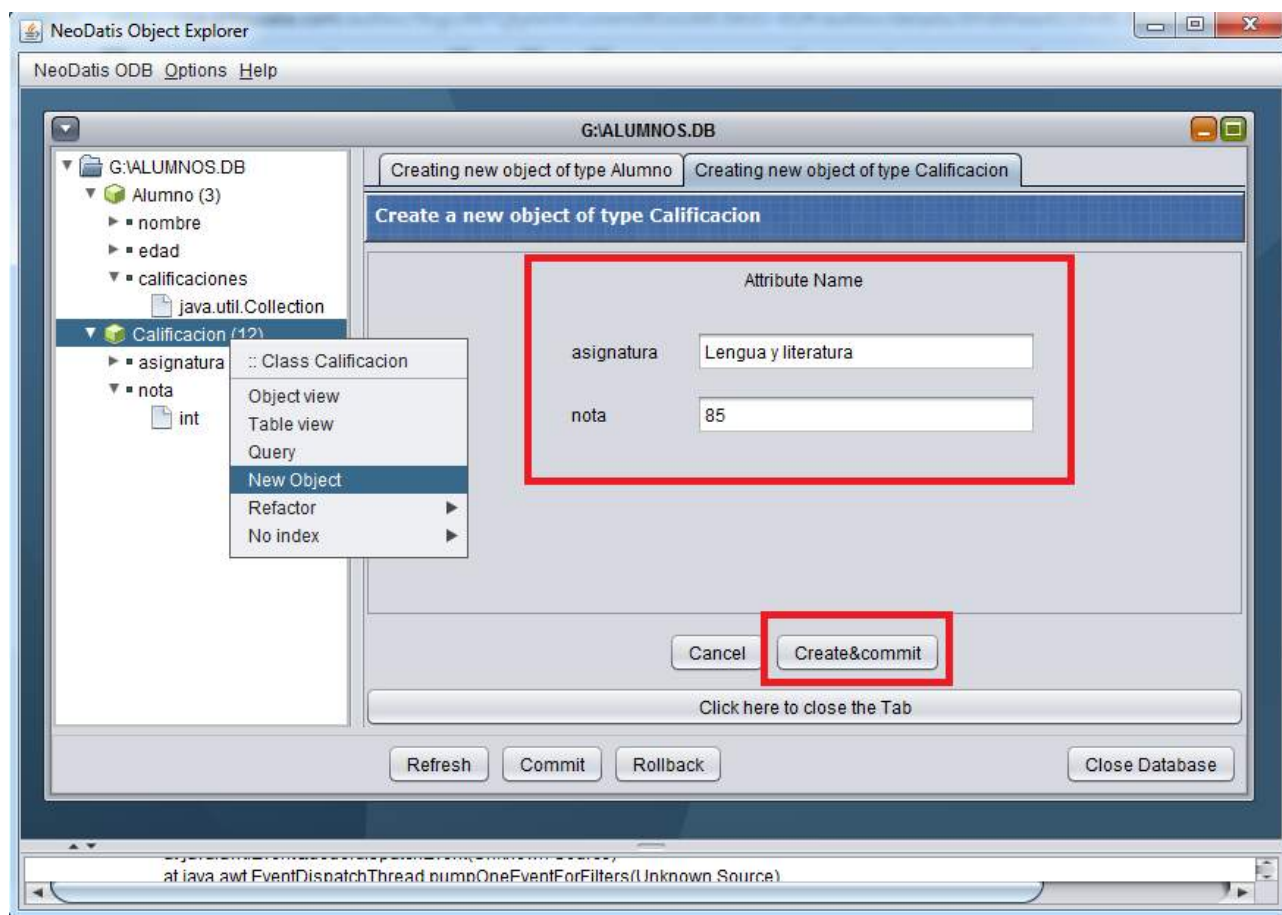
También puedes hacer clic derecho en el nombre de una clase y seleccionar la opción *Query* para localizar los objetos que cumplan un criterio de búsqueda. En el ejemplo de la imagen, queremos localizar al alumno con *nombre* = "Pedro".



Al hacer clic en el botón *Execute* obtendrás el resultado de la búsqueda.

## New Object

Incluso es posible añadir nuevos objetos en la base de datos, haciendo clic derecho en el nombre de la clase y seleccionando la opción *New Object*.



# Consultar BD Alumnos

## Consultar base de datos de alumnos en Java

En este apartado crearemos un **programa Java que permita leer secuencialmente los objetos almacenados** en la base de datos *ALUMNOS.DB*.

Dentro del mismo proyecto, crea otra clase con método *main* con el nombre *RecuperarObjetos*. Copia y pega el código siguiente; más tarde lo analizaremos detenidamente.

```
import org.neodatis.odb.ODB;
import org.neodatis.odb.ODBFactory;
import org.neodatis.odb.Objects;

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

        ODB objAlumnos = ODBFactory.open("G:/ALUMNOS.DB");

        Objects<Alumno> alumnos=objAlumnos.getObjects(Alumno.class);

        Alumno alu;
        while (alumnos.hasNext()) {
            alu=alumnos.next();
            System.out.println(alu.getNombre() + " - " + alu.getEdad() + "
años");
            for (Calificacion c : alu.getCalificaciones()) {
                System.out.println("    " + c);
            }
        }

        objAlumnos.close();
    }
}
```

En primer lugar, nos centraremos en esta línea:

```
Objects<Alumno> alumnos=objAlumnos.getObjects(Alumno.class);
```

Si lo que deseamos es recuperar todos los objetos de la clase *Alumno* almacenados en la base de datos, bastará con ejecutar el método ***getObjects()*** de la clase ***ODB***, pasando como argumento la clase cuyos objetos deseamos recuperar, en este caso *Alumno.class*. El método *getObjects()* devuelve un objeto de la clase genérica ***Objects*** que contiene la colección de objetos recuperados.

Los objetos de la clase *Objects* actúan como cursores que pueden recorrerse hacia adelante con el método *next()*. Para controlar el final de fichero utilizamos el método *hasNext()*, que devuelve *true*, si hay más objetos pendientes de lectura, y *false* en caso contrario.

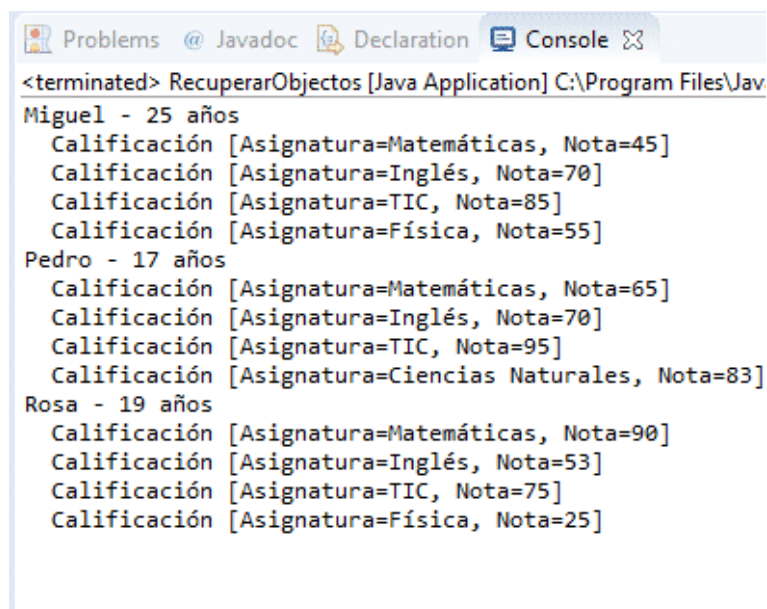
```
alu=alumnos.next();
```

Con esta línea, recuperamos el siguiente objeto *Alumno*. La variable *alu* será la referencia que nos dé acceso al objeto.

```
for (Calificacion c : alu.getCalificaciones()) {  
    System.out.println(" " + c);  
}
```

Sabemos que un objeto *Alumno* contiene objetos *Calificacion*, así que las recorremos con una estructura *for*.

Si has llegado a ejecutar el programa, el resultado habrá sido éste:



```
<terminated> RecuperarObjetos [Java Application] C:\Program Files\Jav  
Miguel - 25 años  
  Calificación [Asignatura=Matemáticas, Nota=45]  
  Calificación [Asignatura=Inglés, Nota=70]  
  Calificación [Asignatura=TIC, Nota=85]  
  Calificación [Asignatura=Física, Nota=55]  
Pedro - 17 años  
  Calificación [Asignatura=Matemáticas, Nota=65]  
  Calificación [Asignatura=Inglés, Nota=70]  
  Calificación [Asignatura=TIC, Nota=95]  
  Calificación [Asignatura=Ciencias Naturales, Nota=83]  
Rosa - 19 años  
  Calificación [Asignatura=Matemáticas, Nota=90]  
  Calificación [Asignatura=Inglés, Nota=53]  
  Calificación [Asignatura=TIC, Nota=75]  
  Calificación [Asignatura=Física, Nota=25]
```

Resultado final.



# Importar y exportar en Neodatis

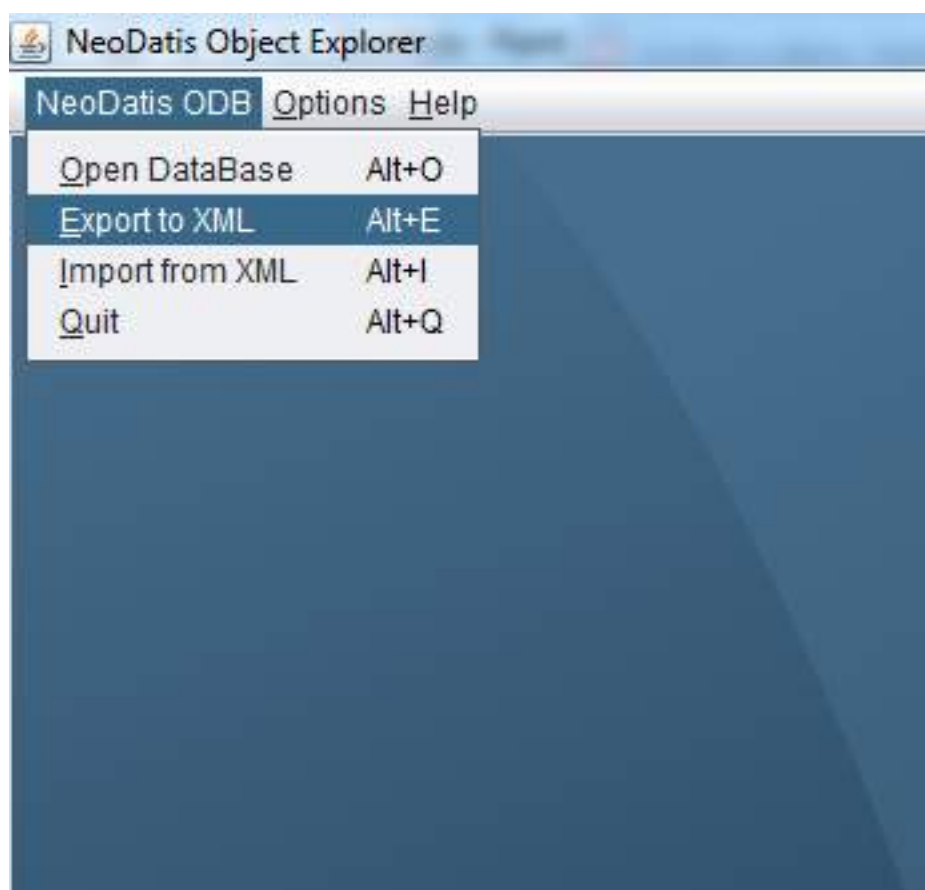
## Exportar Object DB a XML

Dentro de la interfaz de usuario de NeoDatis es posible **exportar la base de datos a formato XML**.

De esta forma tendremos una copia de seguridad de los datos.

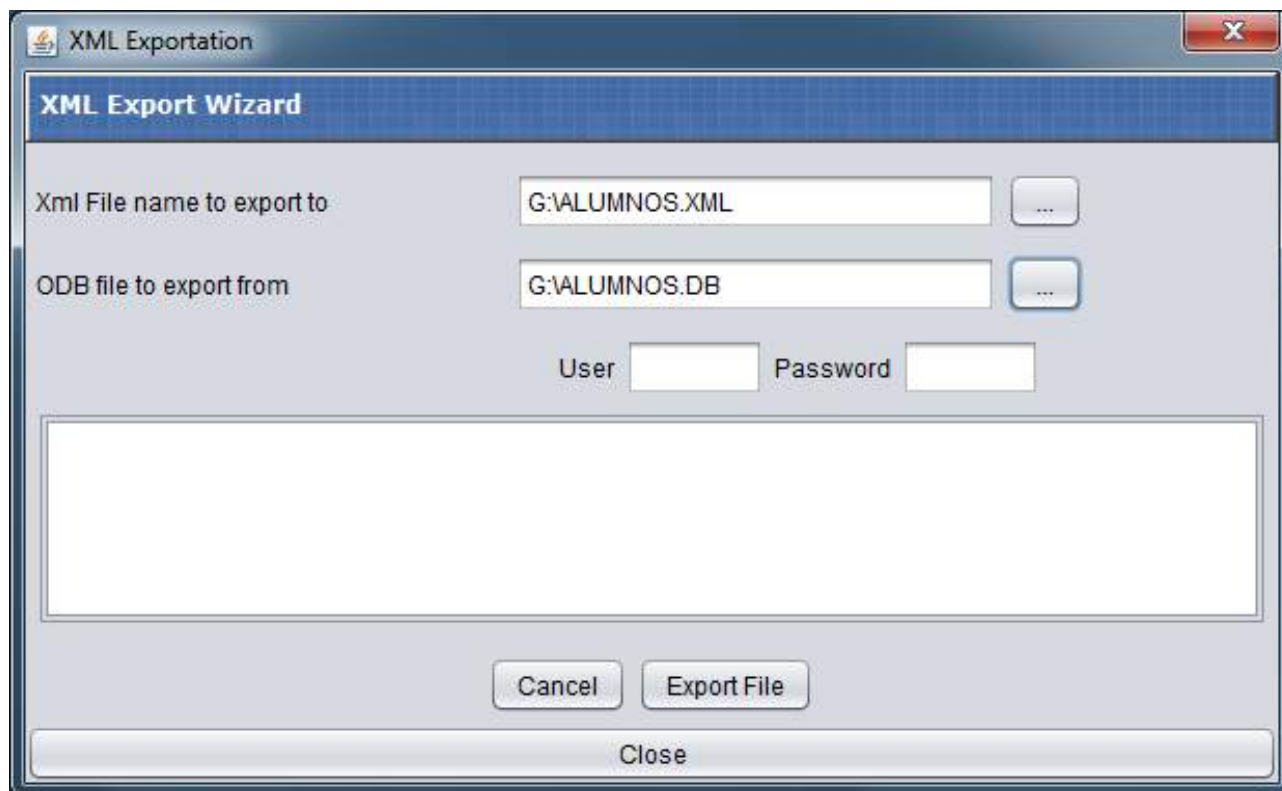
### ¿Quieres ponerlo en práctica?

Abre la interfaz de usuario de NeoDatis y selecciona en el menú *NeoDatis ODB / Export to XML*.

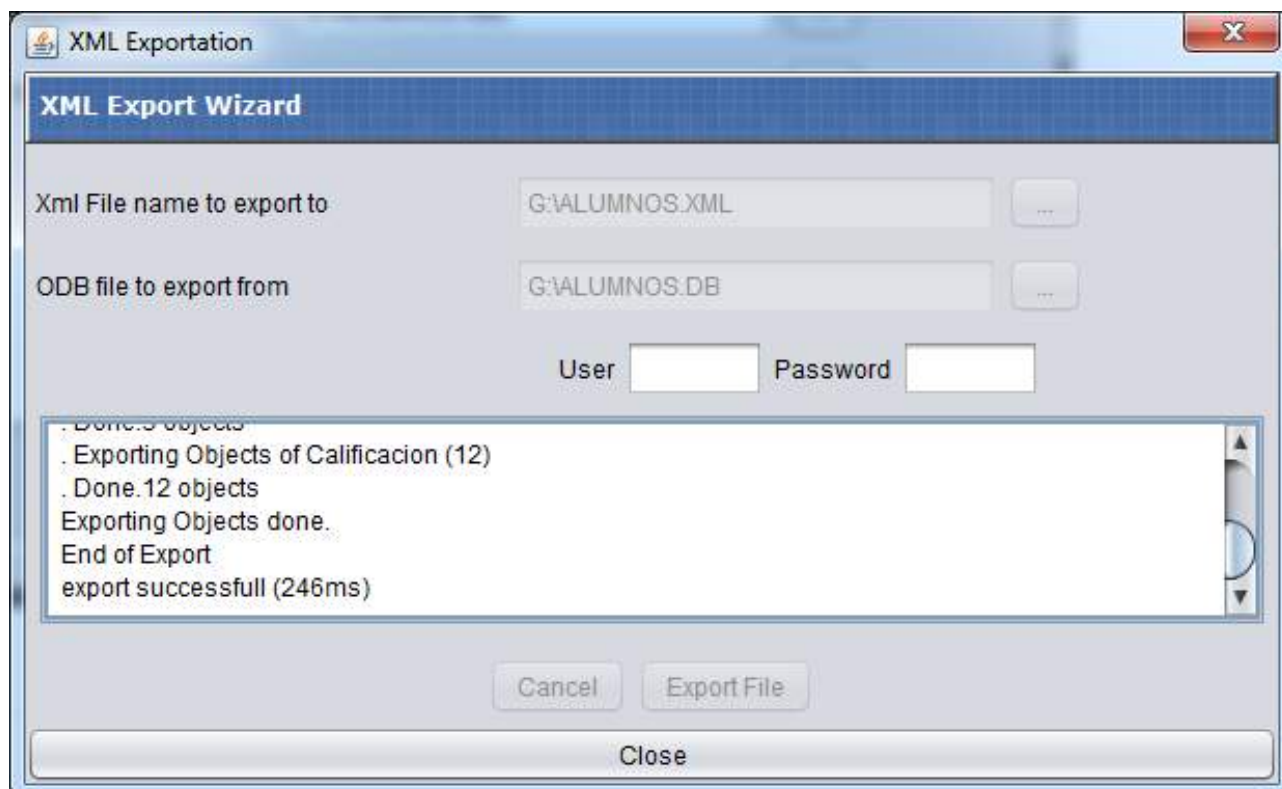


Dentro del cuadro de diálogo *XML Exportation*, debes rellenar dos campos:

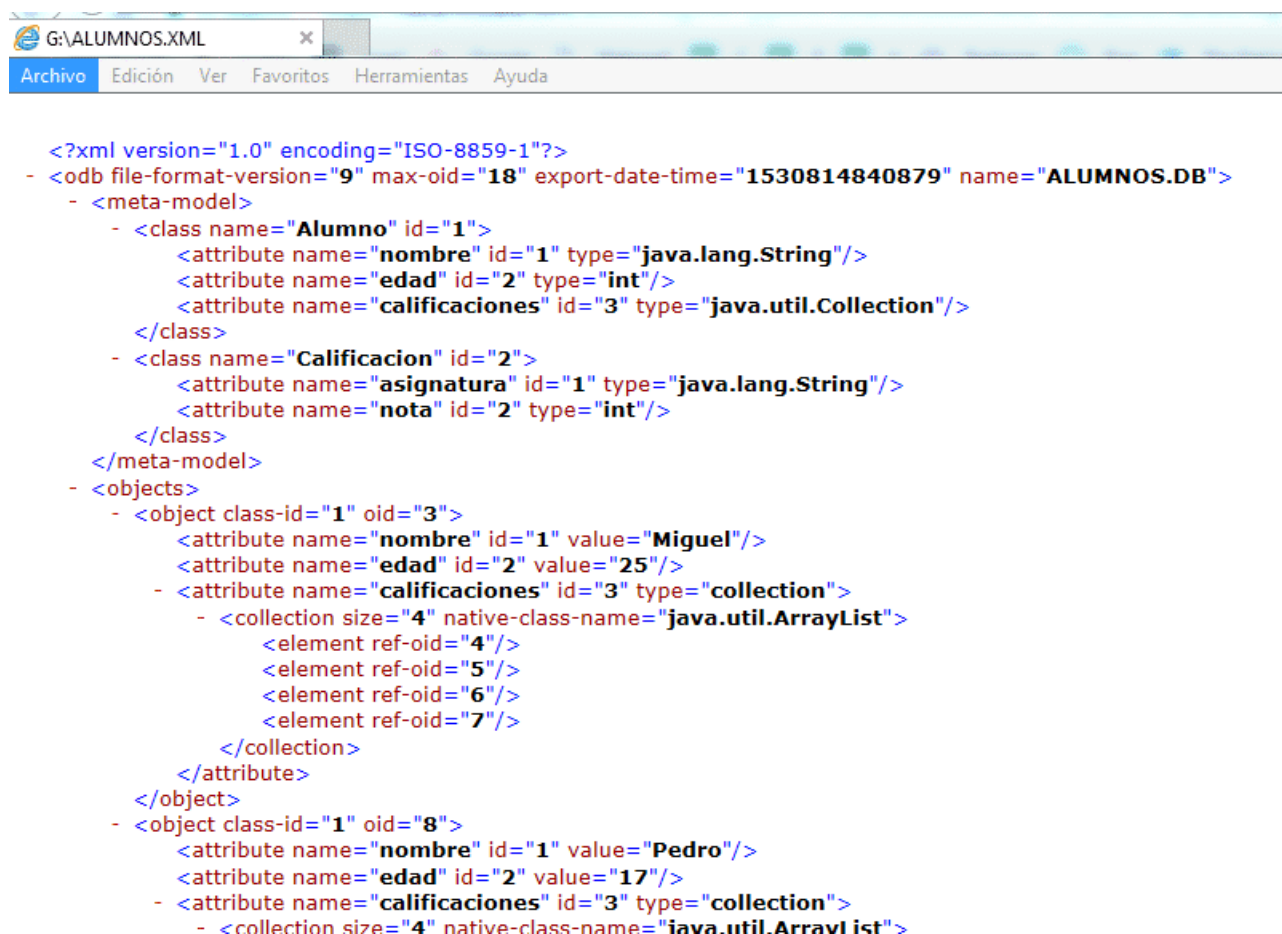
- ***Xml File name to export to:*** escribe aquí la ruta y nombre del futuro fichero XML de destino.
- ***ODB file to export from:*** escribe aquí la ruta y nombre del archivo que contiene la base de datos de origen.



Finaliza haciendo clic en el botón *Export File*. Si la exportación se ha efectuado con éxito, en la consola de resultados verás el mensaje final *export successfull*.



Ya puedes hacer clic en el botón *Close*. El archivo XML que has obtenido tendrá el siguiente formato:



Como puedes comprobar, el documento XML contiene, tanto la estructura de las clases, como los datos guardados en los objetos. Si se llegara a deteriorar el archivo *ALUMNOS.DB* podríamos volver a construirlo a partir del documento XML, ejecutando la opción *Import from XML* en el menú de la interfaz de usuario de NeoDatis.

Ya hemos visto cómo se exporta un objeto DB a XML. Veamos ahora cómo se importa.

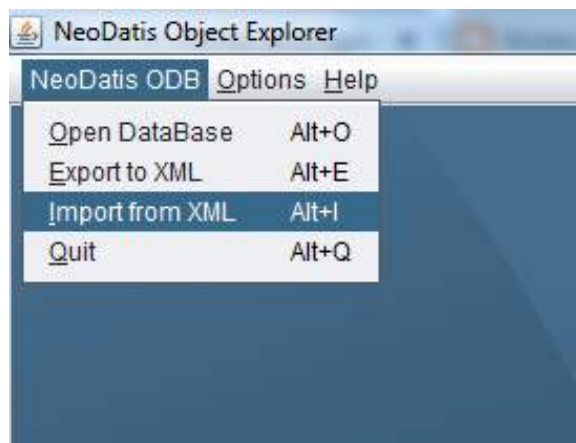
## Importar XML para obtener Object DB

En este apartado realizarás la **importación de una copia de seguridad de una base de datos almacenada en formato XML**. La base de datos contendrá el inventario de un almacén de productos de alimentación.

Para este ejemplo, necesitas el archivo "almacen.xml" que puedes descargar de la versión online de esta lección.

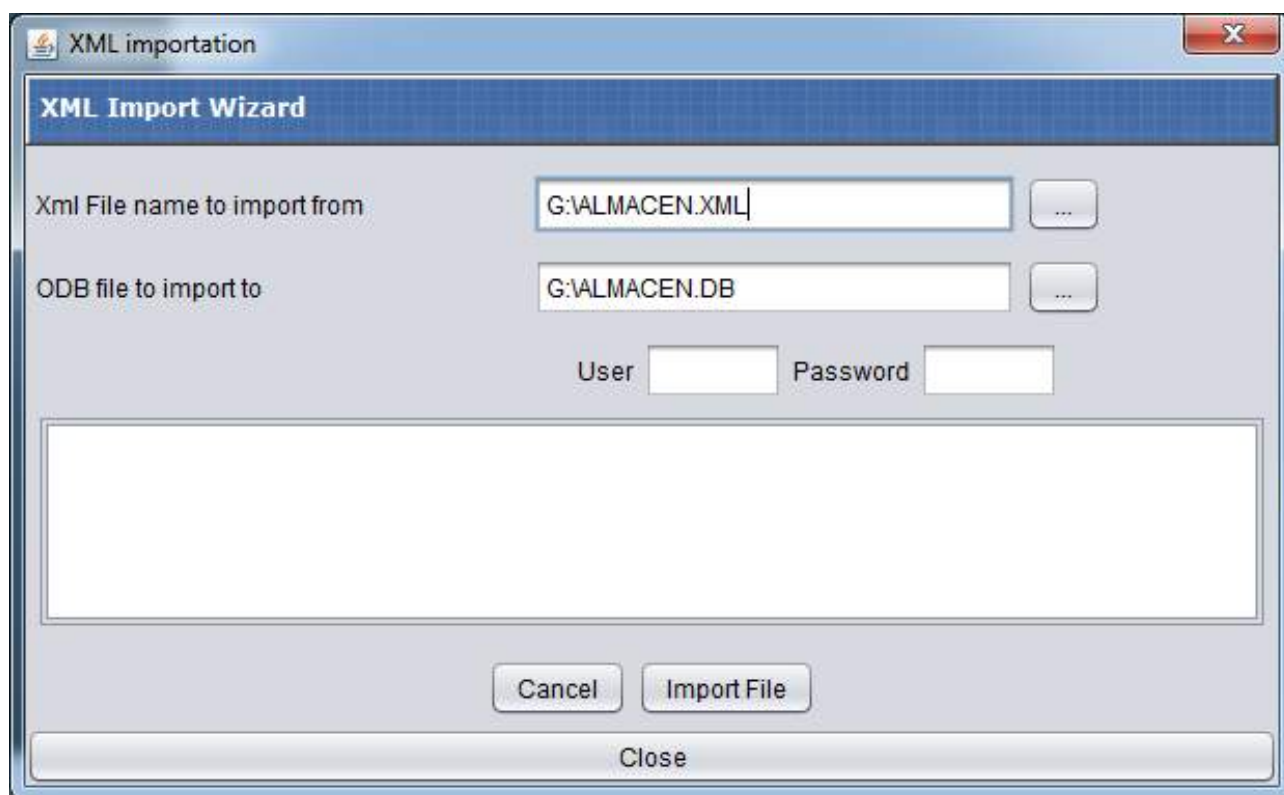
Si ya has realizado la descarga del archivo, pongámonos manos a la obra para importarlo y obtener la base de datos orientada a objetos.

Abre la interfaz de usuario de NeoDatis y selecciona en el menú **NeoDatis ODB / Import from XML**.



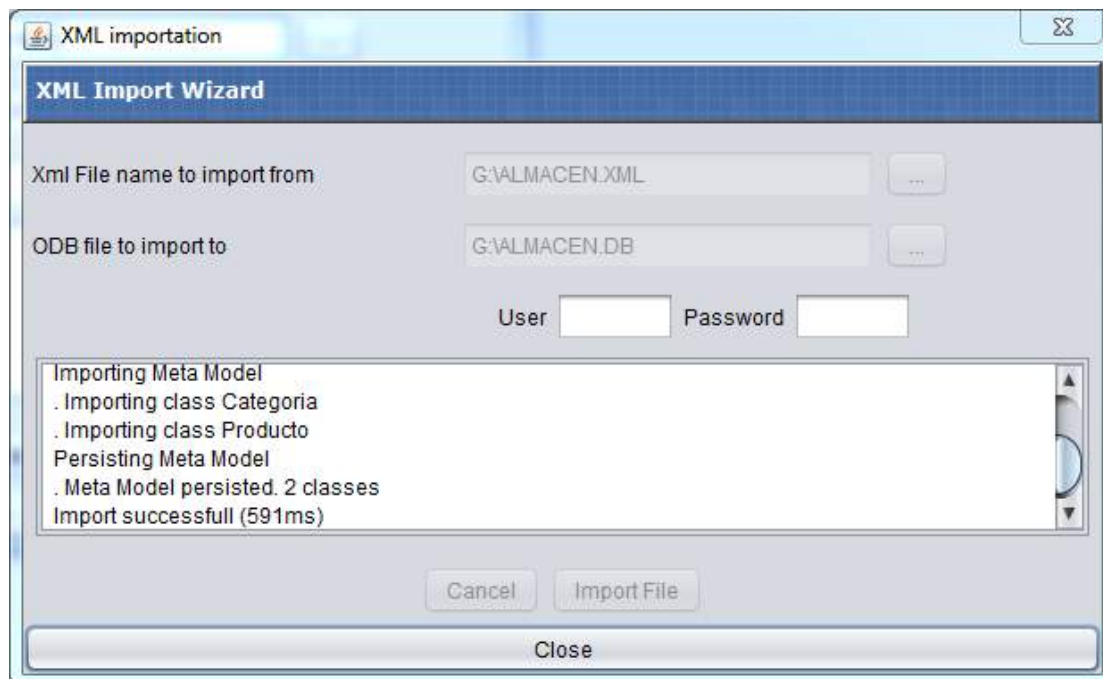
Dentro del cuadro de diálogo **XML importation** tendrás que rellenar dos campos:

- **Xml File name to import from:** escribe aquí la ruta y nombre del archivo XML que has descargado. Puedes utilizar el botón de la derecha, con tres puntos, para seleccionar la carpeta y el archivo.
- **ODB file to import to:** escribe aquí la ruta y nombre de la nueva base de datos que será el destino de la importación.



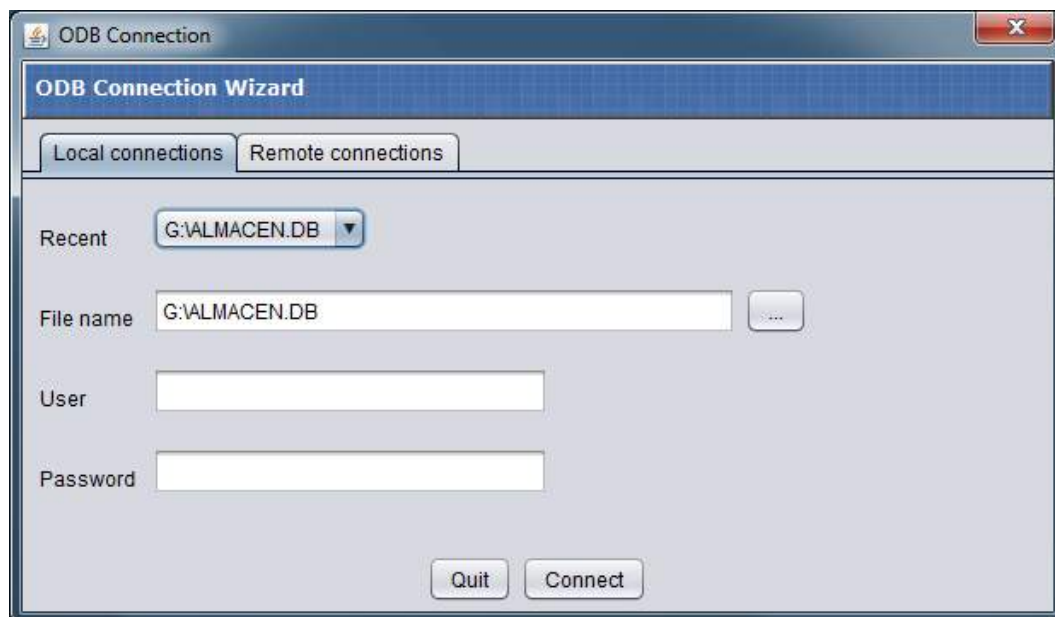
Por último, haz clic en el botón **Import File**.

Si la importación se ha efectuado con éxito, en la consola de resultados verás el mensaje final *Import successfull*. Ya puedes hacer clic en el botón **Close** para cerrar el cuadro de diálogo.

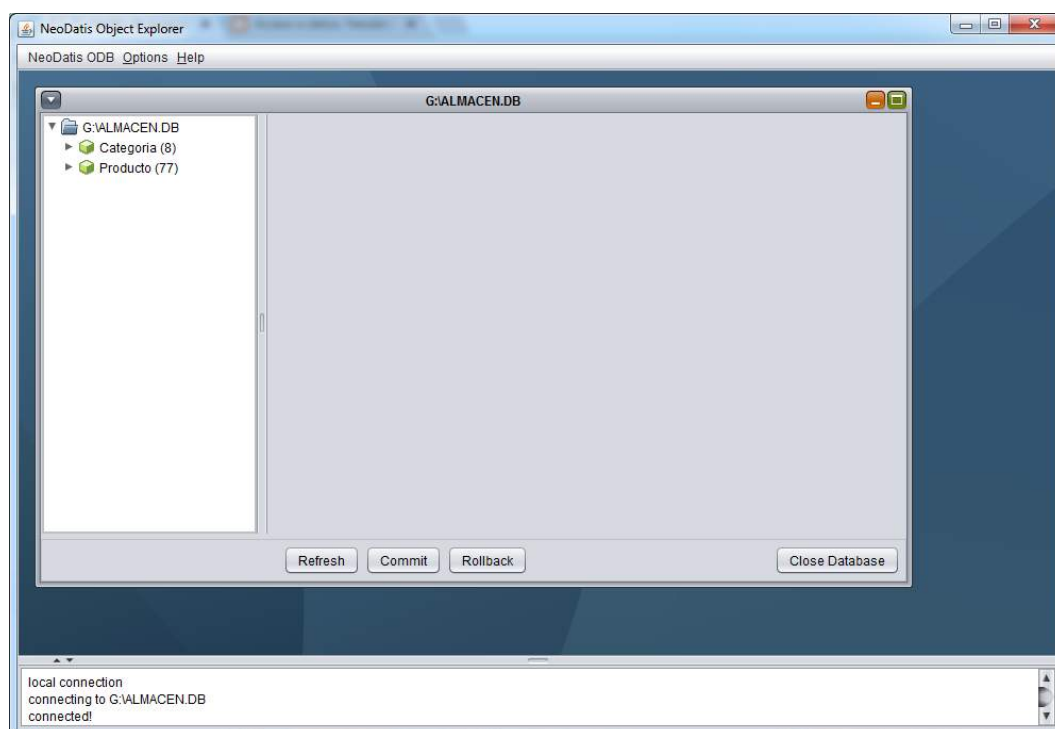


Para comprobar que la importación ha sido un éxito, puedes abrir la base de datos, explorar y familiarizarte con el contenido.

Selecciona en el menú **NeoDatis ODB / Open Database**. Escribe la ruta y nombre del nuevo archivo de base de datos en el campo **File name**.



Para terminar, haz clic en el botón **Connect**.



La interfaz de usuario de NeoDatis te está indicando que en la base de datos hay almacenados ocho objetos de la clase *Categoria* y, setenta y siete objetos de la clase *Producto*. Existe una asociación entre ellos, ya que cada objeto *Categoria* está compuesto por los atributos *nombre* y *productos* (colección de objetos *Producto*).



# Despedida

## Resumen

Has terminado la lección, repasemos los puntos más importantes que hemos tratado.

- Dentro de la carpeta de la aplicación NeoDatis se encuentra el archivo ***neodatis-odb-1.9.30.689.jar*** que contiene las librerías de clases necesarias para formar parte de la capa de persistencia de nuestras aplicaciones Java. Haciendo uso de dichas clases, grabar y recuperar objetos es una tarea muy sencilla.
- Desde la interfaz de usuario de NeoDatis podemos **examinar los objetos almacenados en las bases de datos** que hemos creado previamente desde nuestra aplicación Java.
- La interfaz de usuario de NeoDatis nos permite crear **copias de seguridad de la base de datos en formato XML** por medio de la opción de menú ***Export to XML***. Si la base de datos original llegara a deteriorarse, podríamos restaurarla importando los datos desde la copia XML a través de la opción de menú ***Import from XML***.