



## **ACCESO A DATOS**

### **UD4. Bases de Datos Orientadas a Objetos**

#### **Práctica 1 (voluntaria)**



Reconocimiento – No Comercial: permite el uso de la obra siempre que no sea con fines comerciales y se reconozca la autoría de su creador. Se autoriza la reproducción realizada bajo demanda para alumnos cursando Formación Profesional en el IES Ciudad Escolar.

## Contenido

<b>Objetivos:</b> .....	2
<b>Materiales y recursos:</b> .....	2
<b>Desarrollo de la actividad:</b> .....	2
<b>Entrega:</b> .....	3

## **UD4. Bases de Datos Orientadas a Objetos**

### **Práctica 1 (voluntaria)**

#### **Objetivos:**

- Poner en práctica los recursos estudiados en clase para la persistencia de datos haciendo uso de Bases de Datos Orientadas a Objeto desde aplicaciones JAVA.
- Asimilar y poner en práctica las buenas prácticas de programación (evitando “hardcodes” y la gestión ineficiente de recursos).

#### **Materiales y recursos:**

- Ejercicios realizados en clase incluidos en la unidad didáctica.
- Ordenador del/a alumno/a.
- Fichero de configuración del log: logback.xml
- Software: IDE Eclipse.
- Librerías JAVA: db4o.jar, logback-classic-1.4.11.jar, logback-core-1.4.11.jar, slf4j-api-2.0.7.jar

#### **Desarrollo de la actividad:**

Desarrolla una aplicación en JAVA que permita guardar, consultar, borrar y modificar una instancia de la clase Instituto en una BD OO db4o.

La clase Instituto debe incluir los siguientes atributos:

- Una cadena de texto que almacene el nombre del instituto.
- Un número identificador del centro.

- Una lista de alumnos matriculados. La clase Alumno utilizada en proyectos previos te servirá.

#### Requisitos:

- Debes implementar, entre otros métodos y constructores, los siguientes métodos en la clase Instituto:

**public void matricularAlumno(Alumno al):** añadir a la colección

**public void expulsarAlumno(Alumno al):** sacar de la colección

**public boolean cambiarIdInstituto(int nuevold)**

- Debes implementar, entre otros métodos y constructores, los siguientes métodos en la clase Bbddoo:

**public void guardarAlumno(Alumno al)**

**public List<Alumno> getTodosAlumnos(String nombreInstituto)**

**public boolean guardarInstituto(Instituto insti)**

**public Instituto getInstituto(Instituto institutoBuscado)**

**public boolean borrarInstituto(Instituto institutoABorrar)**

¿Qué pasa con los alumnos matriculados?

**public boolean expulsarATodosAlumnos (String nombreInstituto):**

este método debe borrar de la BD transaccionalmente todos los alumnos matriculados

**public Instituto consultaMatriculasInstituto(String idInstituto)**

**public Instituto consultaInstiMatriculado(String nomAlumno)**

**Nota:** La BD soporta operaciones en cascada. Por ejemplo, para borrar los alumnos matriculados en un instituto durante el borrado del instituto, podemos indicar... ¡Pruébalo!

```
db.ext().configure().objectClass(Instituto.class).cascadeOnDelete(true);
```

#### Entrega:

No requiere entrega.