BDOO

Introducción

- **BDOO**: Bases de datos orientadas a Objetos
- Son aquellas bases de datos cuyo modelo de datos está orientado a objetos. Almacenan tanto métodos como datos
- Las BDOO simplifican la programación orientada a objetos (POO) almacenando directamente los objetos en la BD.
- **SGBDOO**: sistema gestor de bases de datos que almacena objetos.
- **ODMG:** (Object Database Management Group) es un grupo de fabricantes de bbdd que tiene como objetivo definir estándares para los SGBDOO. Uno de dichos estándares (se llama ODMG, precisamente) especifica los elementos que debe contener el SGBDOO

El estándar ODMG

- La última versión del estándar, ODMG 3.0 propone los siguientes componentes para un SGBDOO:
  - Modelo de objetos
  - Lenguaje de definición de objetos (**ODL**, Object Definition Language)
  - Lenguaje de consulta de objetos (**OQL**, Object Query Language)
  - Conexión con los lenguaje C++, Smalltalk y Java

ODL

- El lenguaje ODL es el equivalente al lenguaje de definición de datos (DDL) de los SGDB tradicionales.
- Define atributos, relaciones y signatura de las operaciones
- La sintaxis de ODL extiende el lenguaje de definición de interfaces de CORBA (Common Object Broker Architecture).
- Algunas de las palabras reservadas para definir los objetos son:
  - **class** → declaración del objeto
  - **extent** → define la extensión, nombre para el actual conjunto de objetos de la clase.
  - **key[s]** → declara lista de claves
  - **attribute** → declara un atributo
  - ….

```
class Cliente (extent Clientes key NIF)
{
	attribute struct Nombre_Persona {
			string apellidos,
			string nombrepern} nombre;
	attribute string NIF;
	attribute date fecha_nac;
	attribute enum Genero {H,M} sexo;
	attribute struct Direccion{
		string calle,
		string poblac} direc;
	attribute set<string> telefonos;
 /* Definición de operaciones*/
	short edad();
}
```

El lenguaje de consultas OQL

- **OQL: Object Query Language** es el lenguaje estándar de consultas de BDOO.
- Es, lógicamente, orientado a objetos y está basado en el modelo ODMG. Su sintaxis es similar a SQL.
- No incluye instrucciones de modificación. Para modificar un objeto de la bbdd se utiliza los propios métodos del objeto.
- La sintaxis básica es:

```
SELECT <lista de valores>
FROM <lista de colecciones y miembros típicos>
[WHERE <condición>]
```

Veamos algún ejemplo:

- Obtener el nombre de los clientes que son mujeres:

```
SELECT x.nombre.nombreper FROM x IN Clientes WHERE x.sexo=”M”;
```

- O, la misma consulta, pero utilizando una sintaxis todavía más parecida a SQL:

```
SELECT x.nombre.nombreper FROM Clientes x  WHERE x.sexo=”M”;
```

- Obviamente, OQL admite operadores de comparación, funciones de comparación (IN, LIKE), funciones de agregación (SUM, AVG, MIN, MAX, COUNT) y todo un largo etc.

Ejemplo de BDOO

- A continuación veremos una base de datos sencilla que aporta una API simple. Se trata de **NeoDatis Object Database**
- Desde la web http://neodatis.wikidot.com/ tendremos acceso a las librerías y a la documentación. Tenéis también las librerías en el moodle.
- El ejemplo que se presenta a continuación almacena objetos Jugadores en la base de datos de nombre *neodatis.test*. Para abrir la base de datos se usa la clase **ODBFactory** con el método **open()** que devuelve un **ODB** que es la interfaz principal
- Para almacenar los objetos se usa el método **store()** y para recuperarlos **getObjects().** Por último, para validar los cambios en la bbdd, se usa el método **close()**

```
import org.neodatis.odb.ODB;
import org.neodatis.odb.ODBFactory;
import org.neodatis.odb.Objects;
class Jugadores {                          //Clase Jugadores
        private String nombre;
        private String deporte;
        private String ciudad;
        private int edad;

        public Jugadores() {}
        public Jugadores(String nombre, String deporte,
                String ciudad, int edad) {
        this.nombre = nombre;
        this.deporte = deporte;
        this.ciudad = ciudad;
        this.edad = edad;
        }

        public void setNombre(String nombre) {this.nombre = nombre;}
        public String getNombre() {return nombre;}
        public void setDeporte(String deporte) {this.deporte = deporte;}
        public String getDeporte() {return deporte;}
        public void setCiudad(String ciudad) {this.ciudad = ciudad;}
        public String getCiudad () {return ciudad;}
        public void setEdad(int edad) {this.edad = edad;}
        public int getEdad() {return edad;}
}
```

```
public class EjemploNeodatis {
        public static void main(String[] args) {
        Jugadores j1 = new Jugadores("Maria", "voleibol", "Madrid", 14);  // Crear instancias para almacenar en BD
        Jugadores j2 = new Jugadores("Miguel", "tenis", "Madrid", 15);
        Jugadores j3 = new Jugadores("Mario", "baloncesto", "Guadalajara", 15);
        Jugadores j4 = new Jugadores("Alicia", "tenis", "Madrid", 14);

        ODB odb = ODBFactory.open("neodatis.test");     // Abrir BD
        odb.store(j1);                                  // Almacenamos objetos
        odb.store(j2);
        odb.store(j3);
        odb.store(j4);
        Objects<Jugadores> objects = odb.getObjects(Jugadores.class);    //recuperamos todos los objetos
        System.out.printf("%d Jugadores: %n", objects.size());

        int i = 1;

        while(objects.hasNext()){                                        // visualizar los objetos
          Jugadores jug = objects.next();
          System.out.printf("%d: %s, %s, %s %n",
                            i++, jug.getNombre(), jug.getDeporte(),
                            jug.getCiudad(), jug.getEdad());
        }
        odb.close(); // Cerrar BD
    }
}
```

Ejercicio1

- Crea la clase *Paises* con dos atributos y sus getter y setter. Los atributos son: *private int id; private String nombrepais;*
- Añade también el método *toString()* para que devuelva el nombre del país: *public String toString() {return nombrepais;}*
- Crea la clase Jugadores (como el ejemplo anterior) y añade el siguiente atributo con sus getter y setter: *private Paises pais;*
- Crea una clase Java (con el método *main()*) que cree una base de datos de nombre EQUIPOS.DB e inserte países y los jugadores de esos países.
- Añade otra clase Java para visualizar los países y los jugadores que hay en la base de datos.