Universidad Nacional de la Patagonia
San Juan Bosco

Facultad de Ingeniería - Sede Trelew

Base de Datos II

Trabajo Práctico Final

Bases de Datos Orientadas a Objetos

- 2014 -

Profesor adjunto: Lic. Gabriel Ingravallo

Jefe Trabajos Prácticos: Lic. Cristian Parise

Alumno: Matías iglesias

Bases de Datos Orientadas a Objetos

A finales de los 80's aparecieron las primeras BDOO, es una base de datos inteligente. Soporta el paradigma orientado a objetos almacenando datos y métodos, y no sólo datos. Está diseñada para ser eficaz, desde el punto de vista físico, para almacenar objetos complejos. Evita el acceso a los datos; esto es mediante los métodos almacenados en ella. Es más segura ya que no permite tener acceso a los datos (objetos); esto debido a que para poder entrar se tiene que hacer por los métodos que haya utilizado el programador.

Diferencias principales entre OODB y RDB

La diferencia entre un tipo de base de datos y el otro radica en la naturaleza con la que se guardan, consultan y manejan los datos. En las bases de datos relacionales los datos están en registros de tablas, que almacenan datos propiamente dichos o referencias a registros de otras tablas. En el modelo relacional el mapeo entre los objetos del modelo de negocio y las tablas debe establecerse y configurarse en tiempo de desarrollo. En cambio usando db4o y las demás bases de datos orientadas a objetos esto es transparente al desarrollador, quien sólo llama desde su código a métodos del objeto que administra la base de datos para realizar una consulta, dar un alta, una modificación o eliminar algún objeto de la base de objetos. El tiempo de desarrollo es mucho menor en una solución orientada a objetos si se utiliza una OODB.

Ventajas principales de OODB frente a RDB

- Mayor velocidad de desarrollo (Tranparencia)
  - No hay mapeos entre objetos y tablas
  - No hay que crear componentes que accedan a las bases de datos
  - El código de acceso a la base es muy sencillo y entendible (métodos get, set y delete) en el caso de db4o.
- Mejor performance con objetos de negocio complejos (árboles, estructuras anidadas, relaciones N a N, relaciones recursivas)
- Fácil Backup (la base completa está en un solo archivo)
- No necesita administración
  - Tiene un recolector de basura - garbage collector - que borra los objetos que no son referenciados
  - Al cambiar algo en las clases no se necesita modificar nada en la base de objetos
- Las búsquedas se hacen directamente usando objetos.
  - Búsquedas usando objetos, sencillas (QBE, "Query By Example")
  - Búsquedas Nativas (la manera recomendada para buscar)
  - Búsquedas usando la "SODA Query API" (búsquedas de bajo nivel)
- Los cambios en los objetos (agregar o quitar atributos a una clase) se aplican directamente en la base, sin tener que migrar datos ni reconfigurar nada.

DB4O

Las claves innovadoras de este producto es su alto rendimiento (sobre todo en modo embebido) y el modelo de desarrollo que proporciona a las aplicaciones para su capa de acceso a datos, el cual propugna un abandono completo del paradigma relacional de las bases de datos tradicionales.

De este modo, tenemos las siguientes consecuencias directas resultantes de este nuevo paradigma:

•Deja de existir un lenguaje SQL de consultas/modificaciones para pasar a crearse sistemas de consulta por métodos delegados y actualización/creación/borrado automático de entidades mediante código compilable.

•Se elimina la necesidad de representar el modelo de datos de la aplicación en dos tipos de esquemas: modelo de objetos y modelo relacional. Ahora el esquema de datos del dominio viene representado por la implementación que se realice del diagrama de clases.

•Se consigue evitar el problema del *Object-Relational Impedance Mismatch* sin sacrificar el rendimiento que los mapeadores objeto-relacionales sufren actualmente para llevar a cabo el mismo objetivo.

La mayor clave del éxito que está teniendo este motor de base de datos frente a otros competidores que han desarrollado tecnologías similares, es que se ha optado por un modelo de licenciamiento idéntico al utilizado por empresas como MySQL: licencia dual GPL/comercial. Es decir, si se quiere desarrollar software libre con esta biblioteca, su uso no conlleva ningún coste por licencia; sin embargo si se desea aplicar a un software privativo, se aplica otro modelo de licenciamiento concreto.

Actualmente este producto funciona como una biblioteca para dos tipos de plataformas de desarrollo: Java y.NET (tanto la implementación de Microsoft como la de Mono).

Consigna

Implementar las siguientes clases en Java

- Juez: propiedades: Nombre, matricula profesional, trayectoria (texto)
- Juzgado: propiedades: Fuero (civil, comercial, laboral, penal), Juez (instancia), domicilio, localidad
- Persona: nombre, apellido, dni, sexo, fecha nacimiento
- Causa: propiedades: Juzgado, Nro. Expediente, imputados (conjunto de Persona), testigos (conjunto de Persona), sentencia (texto, nulo si no tiene todavia)

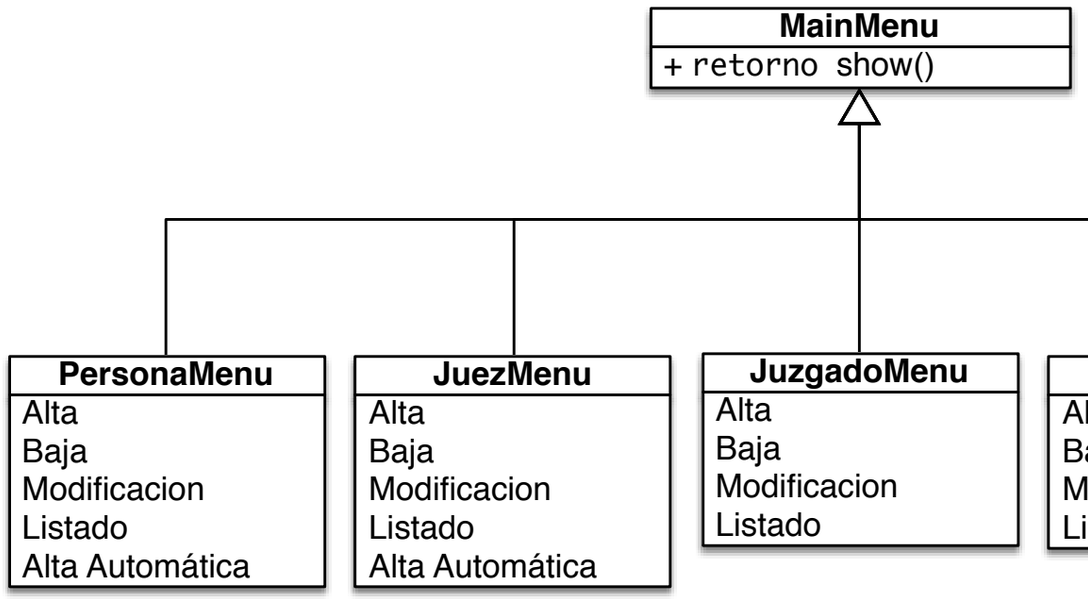
Crear una aplicación Java que genere múltiples instancias de cada una, y utilizando el entonrno eclipse, les dé persistencia en el motor de bases de datos Orientado a objetos: **DB4O**

Hacer las siguientes consultas con QueryByExample

- Mostrar las causas con sentencia que tengan mas de 2 imputados , se deberá mostrar los datos de la causa y los datos de sus imputados
- Mostrar los juzgados del fuero civil que tengan al menos una causa con sentencia y una causa sin sentencia

Resolver también las dos consultas utilizando Native Queries y SODA (una con cada método).

Interfaz




```java
public class PersonaMenu extends MainMenu {
	public retorno show(){
		@SuppressWarnings("resource")
		Scanner scanner = new Scanner (System.in);
		int selection=0;
		int i=0;

		try {
			while(i==0) {
				System.out.println("Seleccione una Opcion:");
				System.out.println("[1] Alta");
				System.out.println("[2] Modificacion");
				System.out.println("[3] Baja");
				System.out.println("[4] Listar");
				System.out.println("[5] Alta Automatica");
				System.out.println("[0] Volver al Menu Principal");
				System.out.print("Opcion: ");

				selection = scanner.nextInt();
				//scanner2.close();
				switch (selection){

		case 1:
				return retorno.PersonaAlta;

			case 2:
				return retorno.PersonaModificacion;

			case 3:
				return retorno.PersonaBaja;

			case 4:
				return retorno.PersonaListar;

			case 5:
				return retorno.PersonaAddRandom;

			case 0:
				i=1;
				break;

			default:
				System.out.println("Opcion invalida. Intente de nuevo");

				};
			}

		} catch (Exception e) {
			// Relanzo la excepcion
			throw e;
		}
		return null;
	}
}
```

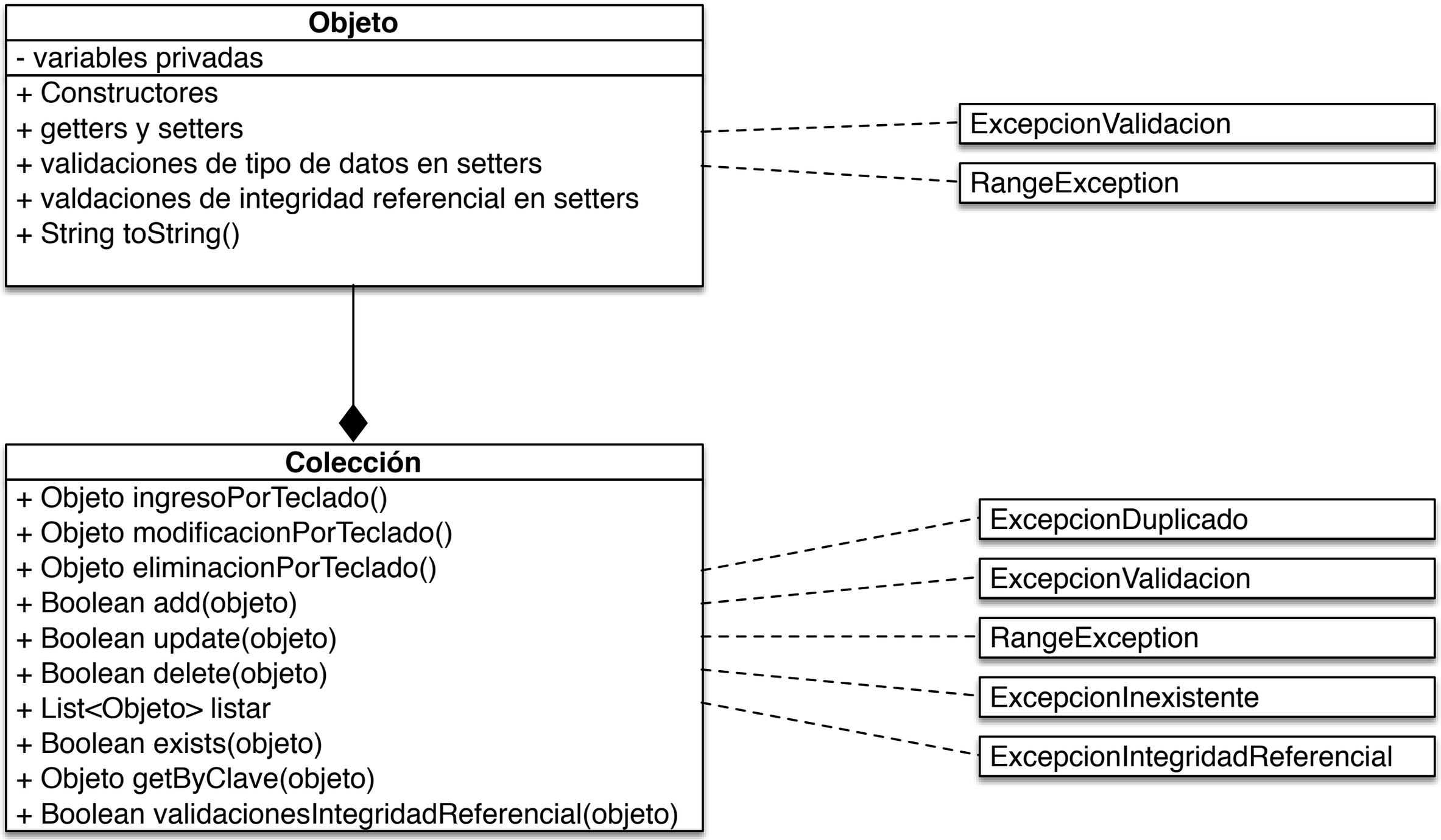
Acceso a Datos

- Concepto de Identidad -> Singleton
- Conexión Persistente
- Indexación de claves de Objetos
- Configuración global

```
package tpFinal_dbo;
import com.db4o.Db4oEmbedded;
import com.db4o.ObjectContainer;
import com.db4o.config.EmbeddedConfiguration;

public class Db {
    private static Db INSTANCE = new Db();
    private static ObjectContainer db;

    private Db() {
      EmbeddedConfiguration configuration = Db4oEmbedded.newConfiguration();
      configuration.common().objectClass(Persona.class).objectField("dni").indexed(true);
      configuration.common().objectClass(Juez.class).objectField("matricula").indexed(true);
      configuration.common().objectClass(Juzgado.class).objectField("numero").indexed(true);
      configuration.common().objectClass(Causa.class).objectField("expediente").indexed(true);
      db = Db4oEmbedded.openFile(configuration, "databaseFile.db4o");
    }

    public static ObjectContainer getConnection () {
      return db;
    }

    public static Db getInstance() {
        return INSTANCE;
    }

    protected void finalize() throws Throwable {
      try{

            System.out.println("Cierro conexion con la Base de Datos");
            db.close();
      } catch(Throwable t){
             throw t;
        } finally{
            super.finalize();
        }
    }
}
```

Más info en http://community.versant.com/documentation/reference/db4o-8.1/java/reference/Content/basics/identity_concept.htm

Jerarquías de Objetos



Query By Example

Causas con sentencia con más de 2 Imputados

```
public void causasConMas2Imputados_QBE() {
     try {
          int cant = 0;
          Db.getInstance();
               //creo un arraylist  de personas imputadas
               ArrayList<Persona> imputados= new ArrayList<Persona>();
               imputados.add(new Persona(null, null, null, null, null));
               imputados.add(new Persona(null, null, null, null, null));

               ArrayList<Persona> testigos= new ArrayList<Persona>();

               Juez juez = new Juez(null, 0, null);

               //creo dos personas y las agrego al arraylist
               Juzgado juzgado = new Juzgado(0, null, juez, null, null);

               Causa causa = new Causa(0, juzgado, imputados, testigos, null);
               causa.setImputados(imputados);

               final ObjectSet<Causa> causas= Db.getConnection().queryByExample(causa);

               System.out.println("Listado de Causas con sentencia que tengan mas de 2 imputados (QBE)");
               System.out.println("-----------------------------------------------------------------------");
               for (Causa c : causas) {
                    if (c.imputados.size()>=2 && c.getSentencia() != null) { //Sino no funca
                         cant++;
                         System.out.println(c);
                    }

               }
               System.out.printf("\nSe encontraron %d casos\n", cant);
               System.out.println("-----------------------------------------------------------------------");

     } catch (Exception e) {
          System.out.printf("ERROR EN EL SISTEMA: %s",e);
     }
}
```

Native Query

Causas con sentencia con más de 2 Imputados

```
public void causasConMas2Imputados_NQ() {
        try {
            Db.getInstance();

            List <Causa> causas = Db.getConnection().query(new Predicate<Causa>() {
                /**
                 *
                 */
                private static final long serialVersionUID = -5928039920141808584L;

                public boolean match(Causa causa) {
                    return (causa.getImputados().size() >= 2 && !causa.getSentencia().equals(null));
                }
            });
            System.out.println("-------------------------------------------------------------");
            System.out.println("Listado de Causas con sentencia que tengan mas de 2 imputados (NQ)");
            System.out.println("-------------------------------------------------------------");
            for (Causa c : causas) {
                System.out.println(c);
            }
            System.out.printf("\nSe encontraron %d casos\n", causas.size());
            System.out.println("-------------------------------------------------------------");


        } catch (Exception e) {
            System.out.printf("ERROR EN EL SISTEMA: %s",e);
        }
    }
```

SODA

Causas con sentencia con más de 2 Imputados

```
public void causasConMas2Imputados_SODA() {
        int cant=0;
        Db.getInstance();

        Query query=Db.getConnection().query();
        query.constrain(Causa.class);


query.descend("sentencia").constrain(null).not().and(query.descend("imputados").descend("size").con
strain(new Integer(2)).greater());

    ObjectSet<Causa> causas=query.execute();
    System.out.println("-----------------------------------------------------------------------");
    System.out.println("Listado de Causas con sentencia que tengan mas de 2 imputados (SODA)");
    System.out.println("-----------------------------------------------------------------------");
    for (Causa c : causas) {
        if (c.imputados.size()>=2) { //Sino no funciona!
            cant++;
            System.out.println(c);
        }
    }
    System.out.printf("\nSe encontraron %d casos\n", cant);
    System.out.println("-----------------------------------------------------------------------");

}
```

Query By Example

Juzgados del Fuero civil con al menos 2 causas con y sin sentencia

```
public void juzgadosFueroCivil_QBE() {
    try {
        int cant=0;
        Db.getInstance();
            Juzgado juzgadoProt = new Juzgado(0, "c", null, null, null);;

            final ObjectSet<Juzgado> juzgados= Db.getConnection().queryByExample(juzgadoProt);

System.out.println("-----------------------------------------------------------------------------------
-------");
            System.out.println("Listado de juzgados del fuero civil con al menos una causa con sentencia y una causa
sin sentencia (QBE)");

System.out.println("-----------------------------------------------------------------------------------
-------");
            for (Juzgado juzgado : juzgados) {
                //Me fijo que tenga causa con sentencia y causa sin sentencia
                if (juzgado.getCantCausasConSentencia()>0 && juzgado.getCantCausasSinSentencia()>0) {
                    cant++;
                    System.out.println(juzgado);
                }
            }
            System.out.printf("\nSe encontraron %d casos\n", cant);

System.out.println("-----------------------------------------------------------------------------------
-------");

        } catch (Exception e) {
            System.out.printf("ERROR EN EL SISTEMA: %s",e);
        }
    }
```

Native Query

Juzgados del Fuero civil con al menos 2 causas con y sin sentencia

```
public void juzgadosFueroCivil_NQ() {
     try {
          Db.getInstance();

               List <Juzgado> juzgados = Db.getConnection().query(new Predicate<Juzgado>() {

                    /**
                     *
                     */
                    private static final long serialVersionUID = 5406016042927709882L;

                    public boolean match(Juzgado juzgado) {
                         return (juzgado.getFuero().equalsIgnoreCase("c") && juzgado.getCantCausasConSentencia() >0 &&
juzgado.getCantCausasSinSentencia() >0 );
                    }
               });

System.out.println("-------------------------------------------------------------------------------------------------");
               System.out.println("Listado de juzgados del fuero civil con al menos una causa con sentencia y una causa sin
sentencia (NQ)");

System.out.println("-------------------------------------------------------------------------------------------------");
               for (Juzgado juzgado : juzgados) {
                    System.out.println(juzgado);
               }
               System.out.printf("\nSe encontraron %d casos\n", juzgados.size());

System.out.println("-------------------------------------------------------------------------------------------------");

          } catch (Exception e) {
               System.out.printf("ERROR EN EL SISTEMA: %s",e);
          }
    }
```

SODA

Juzgados del Fuero civil con al menos 2 causas con y sin sentencia

```
public void juzgadosFueroCivil_SODA() {
    int cant=0;
    Db.getInstance();

    Query query=Db.getConnection().query();
    query.constrain(Juzgado.class);

    query.descend("fuero").constrain("c");

    ObjectSet<Juzgado> juzgados=query.execute();

System.out.println("------------------------------------------------------------------------------------------
---------");
        System.out.println("Listado de juzgados del fuero civil con al menos una causa con sentencia y una causa
sin sentencia (SODA)");

System.out.println("------------------------------------------------------------------------------------------
---------");
        for (Juzgado juzgado: juzgados) {
            if (juzgado.getCantCausasConSentencia()>0 && juzgado.getCantCausasSinSentencia()>0) { //Sino no funca
                cant++;
                System.out.println(juzgado);
            }
        }
        System.out.printf("\nSe encontraron %d casos\n", cant);

System.out.println("------------------------------------------------------------------------------------------
---------");

    }
```

Muchas gracias !

https://github.com/matiasiglesias/db4o