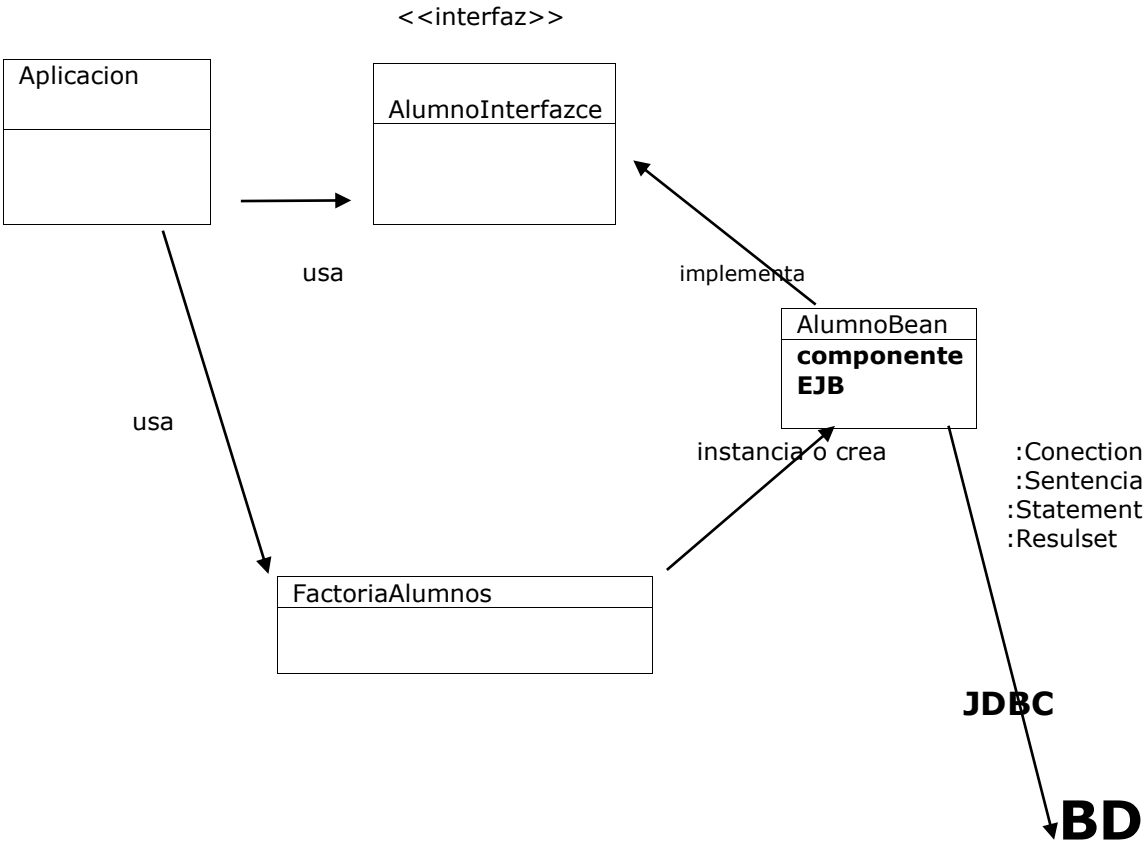
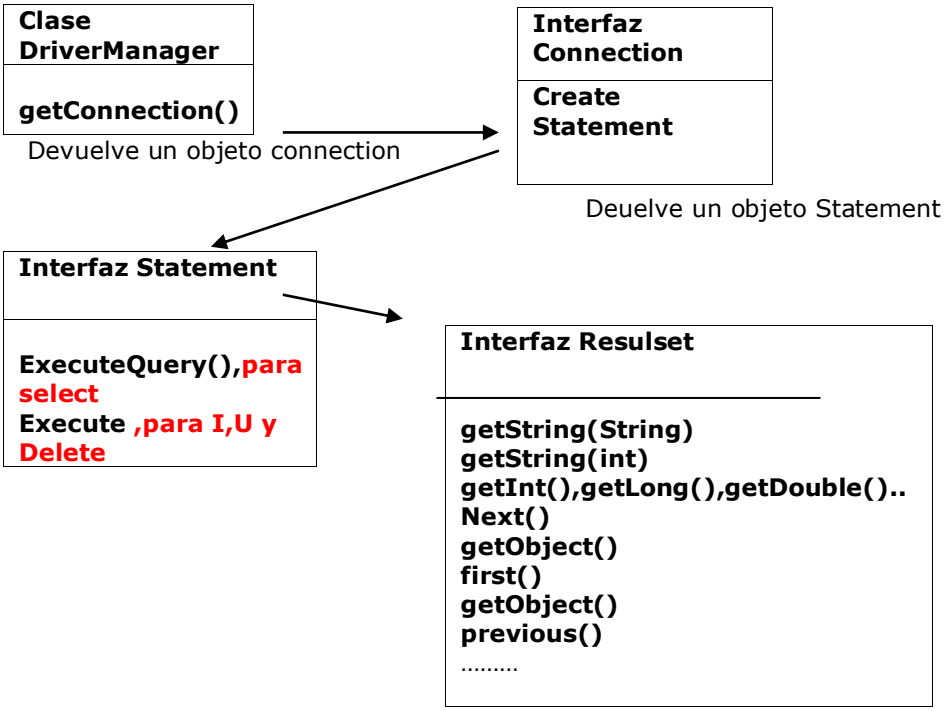
EJERCICIO teórico-práctico JDBC CON ORACLE usando el patrón DAO

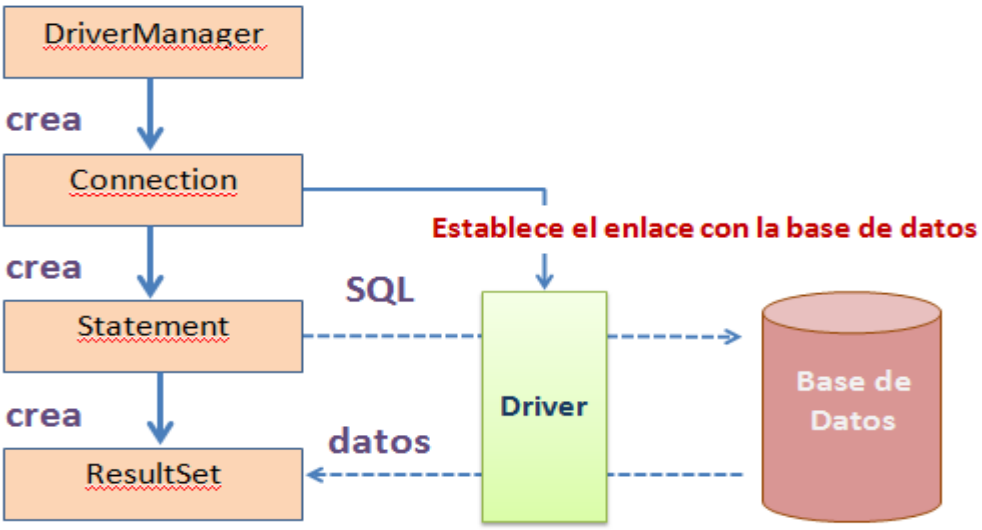
http://www.oracle.com/us/products/tools/dataaccessobject-138824.html

patrón de acceso a datos





ExecuteQuery(),para select, devuelve un objeto resulset que es un cursor
Execute ,para I,U y Delete, no devuelve nada.



Consultar en la ayuda de java las interfaces connection,statement…
http://docs.oracle.com/javase/1.3/docs/api/java/sql/Connection.html
http://docs.oracle.com/javase/1.3/docs/api/java/sql/Statement.html

java.sql

Interface Connection

Method Summary

void	clearWarnings() Clears all warnings reported for this Connection object.
void	close() Releases a Connection's database and JDBC resources immediately instead of waiting for them to be automatically released.
void	commit() Makes all changes made since the previous commit/rollback permanent and releases any database locks currently held by the Connection.
Statement	createStatement() Creates a Statement object for sending SQL statements to the database.
Statement	createStatement(int resultSetType, int resultSetConcurrency) Creates a Statement object that will generate ResultSet objects with the given type and concurrency.
PreparedStatement	prepareStatement(String sql) Creates a PreparedStatement object for sending parameterized SQL statements to the database.
PreparedStatement	prepareStatement(String sql, int resultSetType, int resultSetConcurrency) Creates a PreparedStatement object that will generate ResultSet objects with the given type and concurrency.

TIPOS

SQL	**JAVA**
Char	String
Varchar	String
Integer	int
Number	BigDecimal
….	
.	

MÉTODOS

JAVA	**SQL**
Get	Select
Set	Update
<<Constructor>>	insert
Delete()	delete

Ejercicio teórico práctico.

Dada la base de datos alumno.sql que tiene dos tablas alumno y ciclo

```
drop table alumno;
drop table ciclo;
create table ciclo (codciclo varchar2(3) primary key, denciclo varchar2(70),grado varchar2(8));

insert into ciclo values('DAM','Dearrollo de aplicaciones multiplataforma','superior');
insert into ciclo values('ASIR','Administración de sistemas informáticas y redes','superior');
insert into ciclo values('SMR','Sistemas microinformáticos y redes','medio');
COMMIT;

create table alumno (numexpdte varchar(4) primary key, nombre varchar(25),dni varchar(9),
ciclo varchar(3),fecnac date,
CONSTRAINT ciclo_fk_KEY foreign KEY (ciclo) REFERENCES ciclo (codciclo));

insert into alumno values('1', 'maria gomez', '222222','DAM','01/10/07');
insert into alumno values('2', 'Juan Perez', '9999999','ASIR','01/01/08');
insert into alumno values('3', 'Juan Perez', '9999999','DAM','01/02/08');
insert into alumno values('4', 'Juan Perez', '9999999','DAM','01/03/08');
COMMIT;
```

Es necesario hacer un DAO por cada tabla, vamos a empezar a hacer el DAO para alumno

DISEÑO UML

Diagrama de casos de uso

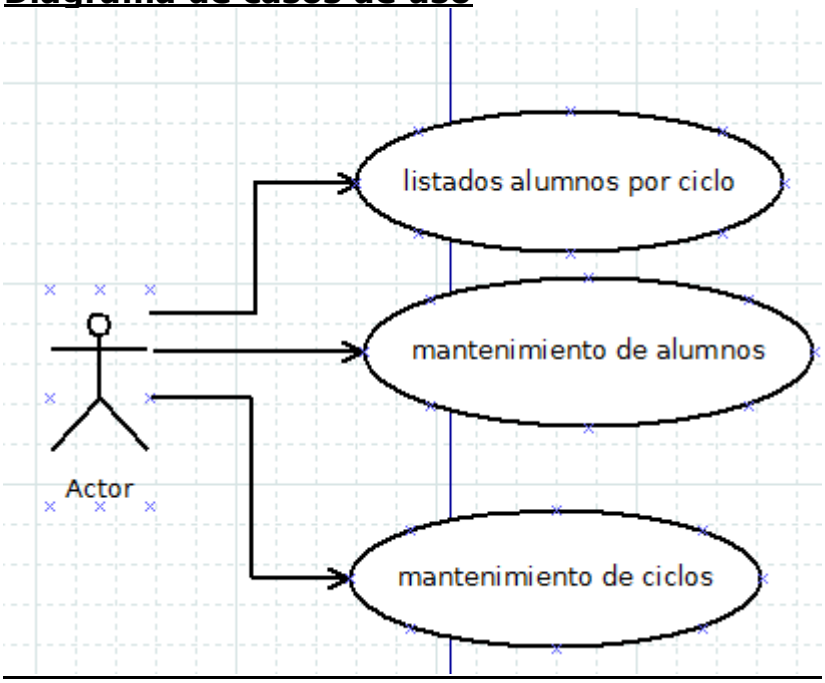
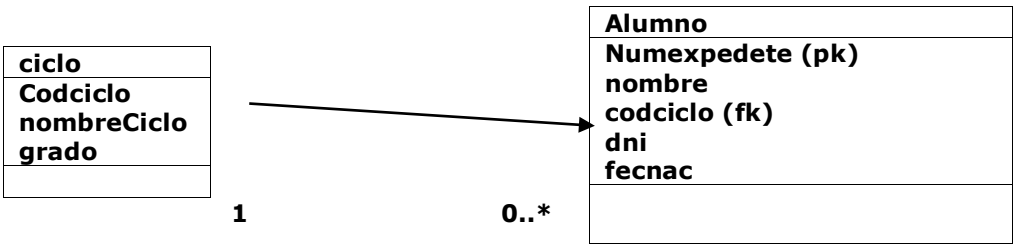


Diagrama de clases



Creamos una clase con el mismo nombre de la tabla y con atributos privados del mismo nombre que los campos. Para acceder a ellos se usan métodos.

Crearemos un EJB por tabla.

Una fila de una BD relacional equivale a un objeto en memoria, para ello necesitamos unos métodos FIND que devuelven objetos de esta clase.

INTERFAZ

Es una relación de prototipos de métodos, se usa porque hay una clase que implementa la interfaz

```
public interface AlumnoInterface {

// metodos GET
    public String getNUMEXPDTE();
    public String getNombre();
    public String getCiclo();
    public String getDni();
    public java.util.Date getFecha();

// metodos SET
    public void  setNombre(String nombre);
    public void  setCiclo(String ciclo);
    public void  setDni(String dni);
    public void  setFecha(java.util.Date fecha);
    //    public void  setNUMEXPDTE(String NUMEXPDTE);//ojo no se puede modificar la clave primaria con la clave primaria habrá que hacer un insert(*)
// métodos FIND
    public AlumnoInterface getAlumnoPorNUMEXPDTE(String NUMEXPDTE);
    public java.util.Collection getAlumnoPorCiclo(String ciclo);
    public java.util.Collection getAlumnoPorNombre(String nombre);

  // METODO BORRADO
    public void delete();

 //necesitamos definir un método para crear un nuevo alumno, que lo vamos a llamar desde los constructores de alumnoBean(*)
    public AlumnoInterface getNuevoAlumno(String NUMEXPDTE,String nombre,String Ciclo, String dni );

}
```

FACTORIA

Clase que solo tiene un método static que se encarga de crear el objeto que implementa la interfaz.
Crea el objeto alumnoBean

```
package proyectodao;

public class FactoriaAlumnos {

    // metodo estatico que devuelve un obj alumnointerfaz

  public static  AlumnoInterface getAlumnoDao(){

        return new AlumnoBean();
    }
}
```

CLASE alumnoBean

Definimos los métodos típicos para hacer acceder a una entidad alumno

Métodos GET … CONSULTAR no tienen parámetros, pero devuelven un tipo de dato que sirva para manejar ese dato en java

```
public String getNUMEXPDTE(){
        return this.NUMEXPDTE;
    }
    public String getNombre(){
        return nombre; // no es obligatorio poner this
    }
    public String getCiclo(){
        return ciclo;
    }
    public String getDni(){
        return dni;
    }
    public java.util.Date getFecha(){
        return fecha;
```

Métodos SET … MODIFICAR ESCRIBIR son siempre void y tienen un parámetro del tipo del atributo.

```
public void  setNombre(String nombre);   //UPDATE
public void  setCiclo(String ciclo);          //UPDATE
public void  setDni(String dni);              //UPDATE
public void  setFecha(java.util.Date fecha);   //UPDATE
 //    public void  setNUMEXPDTE(String NUMEXPDTE);//No se puede
modificar la clave primaria CON LA CLAVE PRIMARIA HABRA QUE HACER
UN INSERT
```

Métodos FIND … métodos que devuelven el objetos que se busca. **SELECT**

```
public AlumnoInterface getAlumnoPorNUMEXPDTE(String NUMEXPDTE);
public java.util.Collection getAlumnoPorCiclo(String ciclo);
public java.util.Collection getAlumnoPorNombre(String nombre);
```

Método Borrado..

```
public void delete();
```

Método para crear un nuevo Alumno

//necesitamos definir un método para crear un nuevo alumno,
//que lo vamos a llamar desde los constructores de alumnoBean
·· INSERT

```
public AlumnoInterface getNuevoAlumno(String NUMEXPDTE,String nombre,String Ciclo,
String dni,java.util.Date fecha);
```

CLASE AlumnoBean

Atributos

```
private String NUMEXPDTE;
private String nombre;
private String ciclo;
private String dni;
private java.util.Date fecha;
private java.sql.Connection conexion=null;
```

métodos

los definidos en la interface

Explicamos cada método.

METODOS GET

```
public String getNUMEXPDTE(){
        return this.NUMEXPDTE;
    }
    public String getNombre(){
        return nombre; // no es obligatorio poner this
    }
    public String getCiclo(){
        return ciclo;
    }
    public String getDni(){
        return dni;
    }
    public java.util.Date getFecha(){
        return fecha;
```

```
//METODO PARA ESTABLECER LA CONEXIÓN
private java.sql.Connection getConexion(){
        //Cargar el driver
        try{
            Class.forName("oracle.jdbc.driver.OracleDriver"); //Obtener el driver
        }catch (ClassNotFoundException e){
            System.out.println("No se encuentra la clase del Driver");
            return null; //devuelve null si va mal
        }

        Connection conexion= null;
        try {

conexion=DriverManager.getConnection("jdbc:oracle:thin:@192.168.33.228:1521:orcl","blanca"
,"blanca");
        } catch (SQLException e){
            System.out.println("No se puede obtener la conexión");
            return null;
        }
        return conexion;
    }
// DriverManager es una clase abstracta
```

MÉTODOS SET

```
public void  setNombre(String nombre){
conexion=getConexion();
        java.sql.Statement sentencia = null;

        try{
            sentencia= conexion.createStatement();
           sentencia.execute(
                    "UPDATE ALUMNO SET NOMBRE='"+nombre+
                    "'WHERE NUMEXPDTE='"+this.NUMEXPDTE+"'");
            sentencia.close();
            conexion.close();
        }catch (SQLException e){
            System.out.println("Error en UPDATE de nombre sobre ALUMNO");
            return;
        }
this.nombre=nombre;

/ / como el nombre del parámetro es igual al del objeto se pone  this al objet o para
diferenciarlo


    }
```

```
public void  setCiclo(String ciclo){

       conexion=getConexion();
       java.sql.Statement sentencia = null;
        try{
            sentencia= conexion.createStatement(); //se crea la sentencia

                     sentencia.execute(
                    "UPDATE ALUMNO SET CICLO='"+ciclo+
                    "'WHERE NUMEXPDTE='"+this.NUMEXPDTE+"'");
                    //modifico el nombre, con el nombre que nos dan de aquel cuyo
expdiente coincida con el objeto actual
            sentencia.close();
            conexion.close();
        }catch (SQLException e){
            System.out.println("Error en UPDATE de  ciclo sobre ALUMNO");
            return;
        }

        this.ciclo=ciclo;
}
```

```
public void  setDni(String dni){
        conexion=getConexion();
        java.sql.Statement sentencia=null;
         try{
            sentencia= conexion.createStatement();

            sentencia.execute(
                  "UPDATE ALUMNO SET DNI='"+dni+
                  "'WHERE NUMEXPDTE='"+this.NUMEXPDTE+"'");
             sentencia.close();
            conexion.close();
        }catch (SQLException e){
            System.out.println("Error en UPDATE de dni sobre ALUMNO");
            return;
        }catch (Exception e){
            System.out.println(e.getClass());
            return;
        }

        this.dni=dni;
    }
```

```
public void  setFecha(java.util.Date fecha){
        conexion=getConexion();
        java.sql.Statement sentencia=null;
         try{
            sentencia= conexion.createStatement();

            sentencia.execute(
                  "UPDATE ALUMNO SET FECNAC='"+fecha+
                  "'WHERE NUMEXPDTE='"+this.NUMEXPDTE+"'");
            sentencia.close();
            conexion.close();
        }catch (SQLException e){
            System.out.println("Error en UPDATE de fecha sobre ALUMNO");
            return;
        }
        this.fecha=fecha;
    }
```

METODOS FIND (SELECT)

```
public AlumnoInterface getAlumnoPorNUMEXPDTE(String NUMEXPDTE){
      conexion=getConexion();
      java.sql.Statement sentencia = null;
     AlumnoBean alumno= null;   (*)

 try{
        sentencia= conexion.createStatement(); //se crea la sentencia

         alumno=new AlumnoBean();
        //se le da valor ejecutandola
        java.sql.ResultSet resultado;
        resultado = sentencia.executeQuery(                    //executeQuery devuelve
              "SELECT * FROM ALUMNO"+                          ResulSet, que es un
                                                               cursor
               "WHERE NUMEXPDTE='"+NUMEXPDTE+"'");

       while(resultado.next()){

           alumno.NUMEXPDTE=resultado.getString("NUMEXPDTE");
                                                                    //al método
                                                               getString le paso el
                                                               nombre de la columna y
                                                               me devuelve el valor de la
                                                               columna como un String
           alumno.ciclo=resultado.getString("CICLO");
           alumno.dni=resultado.getString("DNI");
           alumno.nombre=resultado.getString("NOMBRE");
           //en java hay dos tipos de Date:
           //java.util.date
           //java.sql.date --> para compatibilizar las fechas entre bbdd
           //getDAte devuelve java.sql.date y el alumno.fecha es de tipò java.util.date y tengo
que hacer una conversión
           alumno.fecha=resultado.getDate("FECHA");
        }
         resultado.close();
         sentencia.close();
         conexion.close();
      }catch (SQLException e){
         System.out.println("Error en SELECT de Alumno por NUMEXPDTE sobre ALUMNO");
         return null;
      }
      return alumno;
  }
```

Creo un objeto de tipo AlumnoBean, para poderlo devolver me posiciono en el siguiente y leo mientras haya datos ahora valdría sin el while porque solo hay un elemento
Todos los métodos get de resultSet están sobre cargados permiten pasarle, la columna o el nombre de la columna
(*) bjeto alumno que creo uso com referencia alumnoBean en vez de alumnoInterface, porque si usase alumnoInterface no podría poner alumno.NUMEXPDT tendria que usar un método SET que hace un update.

```
public java.util.Collection getAlumnoPorCiclo(String ciclo){
        conexion=getConexion();
        java.sql.Statement sentencia = null;

        AlumnoBean alumno= null;
        java.util.Collection coleccion =null;

        try{
            sentencia= conexion.createStatement();
            coleccion = new java.util.Vector();                    //Collection no se puede instanciar, pq es un
                                                                   interfaz, utilizamos un Vector, un vector es un obj
                                                                   coleccion
            java.sql.ResultSet resultado;

            resultado = sentencia.executeQuery(
                    "SELECT * FROM ALUMNO"+
                    "WHERE CICLO='"+ ciclo+"'");


           while(resultado.next()){
               alumno=new AlumnoBean();                    Hay que instanciarlo tantas veces como elementos
                                                           de la colección, creo tantos objetos alumnos como
                                                           registros .

               alumno.NUMEXPDTE=resultado.getString("NUMEXPDTE");
               alumno.ciclo=resultado.getString("CICLO");
               alumno.dni=resultado.getString("DNI");
                alumno.nombre=resultado.getString("NOMBRE");
               alumno.fecha=resultado.getDate("FECHA");
               coleccion.add(alumno);
           }
            resultado.close();
            sentencia.close();
            conexion.close();
        }catch (SQLException e){
            System.out.println("Error en SELECT de Alumno por ciclo sobre ALUMNO");
            return null;
        }
        return coleccion;    // Devuelve una colección de alumnos.
    }
```

Con cada Registro:
1º Crear un objeto alumno
2º por cada objeto asignarle las propiedades
3º añadir a este objeto alumno a un objeto colección [coleccion.add(alumno)]

```
public java.util.Collection getAlumnoPorNombre(String nombre){
        conexion=getConexion();
        java.sql.Statement sentencia = null;
        AlumnoBean alumno= null;
        java.util.Collection coleccion =null;

        try{
           sentencia= conexion.createStatement();
           coleccion = new java.util.Vector();

           java.sql.ResultSet resultado;

           resultado = sentencia.executeQuery(
                  "SELECT * FROM ALUMNO"+
                  "WHERE NOMBRE='"+nombre+"'");


          while(resultado.next()){
              alumno=new AlumnoBean();
              alumno.NUMEXPDTE=resultado.getString("NUMEXPDTE");
              alumno.ciclo=resultado.getString("CICLO");
              alumno.dni=resultado.getString("DNI");
              alumno.nombre=resultado.getString("NOMBRE");
              alumno.fecha=resultado.getDate("FECHA");
              coleccion.add(alumno);
          }
           resultado.close();
           sentencia.close();
           conexion.close();
        }catch (SQLException e){
           System.out.println("Error en SELECT de Alumno por nombre sobre ALUMNO");
           return null;
        }
        return coleccion;
    }
```

```
// METODO BORRADO, podemos poner tantos metodos delete como queramos, por ejemplo
   // metodos delete para borrar una coleccion de alumnos

   public void delete() {
      conexion=getConexion();
      java.sql.Statement sentencia=null;
       try{
         sentencia= conexion.createStatement();

         sentencia.execute(
               "DELETE FROM ALUMNO  WHERE NUMEXPDTE='"+this.NUMEXPDTE+"'");
         sentencia.close();
         conexion.close();
      }catch (SQLException e){
         System.out.println("Error  DELETE   sobre ALUMNO");
         return;
      };


      // REInicializamos las var a su valor por defecto, ya que en java no se
      // puede destruir un objeto, por si alguien tiene una ref de este obj
       // aunque lo hayamos borrado de la base de datos , puede parecer que el obj
       //existe, para evitalo hacemos esto.
        this.ciclo=null;
        this.dni=null;
        this.NUMEXPDTE=null;
        this.fecha=null;
        this.conexion=null;
        this.nombre=null;

   }
}
```

INSERT

```
public AlumnoInterface getNuevoAlumno(String NUMEXPDTE,String nombre,String Ciclo,
String dni,java.util.Date fecha){

        conexion=getConexion();
        java.sql.Statement sentencia=null;


        try{
           sentencia=conexion.createStatement();
           sentencia.execute(
                  "INSERT INTO ALUMNO(NUMEXPDTE,NOMBRE,DNI,CICLO,FECNAC VALUES
('"+NUMEXPDTE+"','"+
                  nombre+"','"+dni+"','"+ciclo+"',"+new java.sql.Date(fecha.getTime()")' ");
           }catch(SQLException e){
              System.out.println("Error SQL al insertar Alumno"+ e.getMessage());
                  return null;
           }
        //Instanciamos alumno nuevo y le damos los valores a sus atributos
        AlumnoBean alumno= new AlumnoBean();

        alumno.NUMEXPDTE=NUMEXPDTE;
        alumno.dni=dni;
        alumno.fecha=fecha;
        alumno.nombre=nombre;
        alumno.ciclo=Ciclo;
//devuelvo el objeto alumno
        return alumno;
  }
```

OTRA FORMA DE HACERLO SERÍA CON UN CONSTRUCTOR PARAMETRIZADO

```
/     //para eso me tengo que declarar ese constructor (1)

 return new AlumnoBean(NUMEXPDTE,nombre,ciclo,dni,fecha);
// si hay un error con el insert devuelve un objeto alumno que no existe realmente

//    //(1)
//       AlumnoBean(String NUMEXPDTE,String nombre,String ciclo,String
dni,java.util.Date fecha){
//                  conexion=getConexion();
//                  java.sql.Statement sentencia=null;
//
//
//       try{
//          sentencia=conexion.createStatement();
//          sentencia.execute(
//                 "INSERT INTO ALUMNO(NUMEXPDTE,NOMBRE,DNI,CICLO,FECNAC VALUES
('"+NUMEXPDTE+"','"+
//                  nombre+"','"+dni+"','"+ciclo+"','"+fecha+"'");
//       this.NUMEXPDTE=NUMEXPDTE;
//       this.dni=dni;
//       this.fecha=fecha;
//       this.nombre=nombre;
//       this.ciclo=ciclo;
//          }catch(SQLException e){
//             System.out.println("Error SQL al insertar Alumno"+ e.getMessage());
//
//          }
//       }
```