Implementación del nivel de datos usando db4o

Introducción

En este laboratorio, instalaremos y utilizaremos las clases necesarias para acceder a bases de datos orientadas a objetos db4o..

Objetivos

En este laboratorio realizaremos lo siguiente:

- Instalar las clases necesarias para trabajar con bases de datos db4o
- Abrir y cerrar bases de datos db4o
- Guardar objetos db4o
- Recuperar objetos db4o
- Implementar la lógica del negocio accediendo al nivel de datos

Pasos a seguir

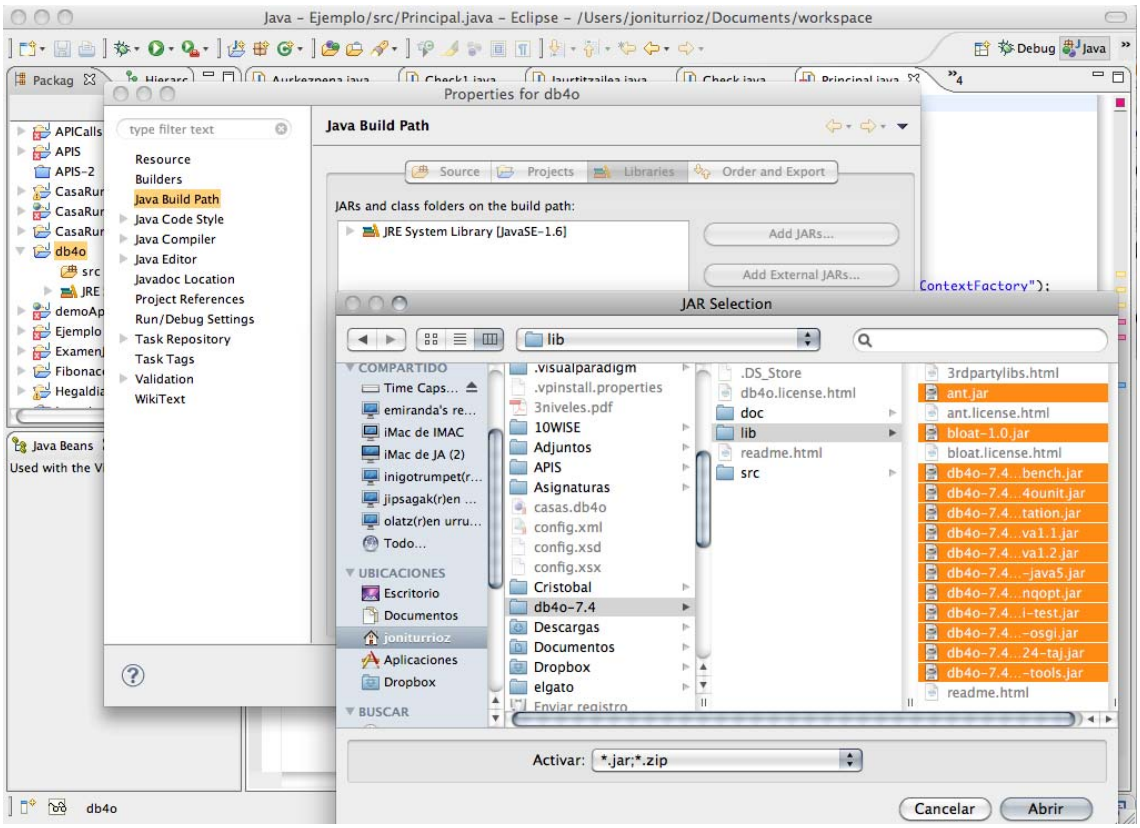
1. Instalación de las clases db4o

a) Descarga del código. Utilizaremos la versión 7.4 que puede encontrarse en la siguiente dirección:
http://download.cnet.com/3001-10254_4-10338175.html?spi=dabf4e06c3cdd490e1e04d134cb7a0bf

b) Descomprimir el fichero .zip en una carpeta.

c) Crear un proyecto Java en la herramienta Eclipse.

d) Cargar las librerías .jar de db40 en el proyecto cargado. Para ello, hay que posicionarse encima del nombre del proyecto, y con el botón derecho del ratón seleccionar “Properties”. En la pantalla que aparece, a la izquierda, seleccionar “Java Build Path”, y a la derecha seleccionar “Libraries” y pulsar en el botón “Add external jar”, tal y como aparece en la siguiente imagen. Para terminar, seleccionar y cargar todos los ficheros .jar que aparecen en la subcarpeta “lib” que se ha descomprimido en el apartado b)



2. Crear una clase Java con un método main.

Crear una clase Java y definir en el método main las llamadas para abrir y cerrar una base de datos, tal y como aparece en el siguiente código.

```
public static void main(String[] args) {
  String DB4OFILENAME="c:\\formula1.db4o";
  ObjectContainer db = Db4o.openFile(Db4o.newConfiguration(),
DB4OFILENAME);
  try {
        //hacer algo
       } finally {
          db.close();
        }
}
```

3. Insertar objetos en la base de datos

Guardaremos objetos de la clase Pilot, cuyo código es el siguiente:

```
public class Pilot {
      String name;
      int points;
      public Pilot(String name, int x){
            this.name=name;
            this.points=x;
      }
      public String  toString(){
            return name+" "+Integer.toString(points);
      }
      public void addPoints(int x){
            this.points=x;
      }
}
```

En la clase principal añadiremos un método para guardar objetos, con el siguiente código:

```
public static void storePilot(ObjectContainer db,String name,int
points) {
        Pilot pilot = new Pilot(name, points);
        db.store(pilot);
        System.out.println("Stored " + pilot);

    }
```

Y añadiremos las siguientes instrucciones dentro de un bloque `try { }` para cerar y guardar 2 pilotos.

```
            storePilot(db,"Michael Schumacher", 100);

            storePilot(db,"Rubens Barrichello", 99);
```

4. Realizar consultas a la base de datos

Para recuperar todos los pilotos definiremos el siguiente método:

```
public static void retrieveAllPilotQBE(ObjectContainer db) {
        Pilot proto = new Pilot(null, 0);
        ObjectSet result = db.queryByExample(proto);
        listResult(result);
    }
```

Que usa el siguiente método para imprimir los resultados:

```
public static void listResult(ObjectSet s){
     Pilot p;
     while (s.hasNext()) {
          p=(Pilot)s.next();
          System.out.println(p);
     }
}
```

Añadir dentro del bloque `try { }` anterior, la llamada al método para realizar la consulta de todos los pilotos.

5. Borrar objetos

En el siguiente método se borra un piloto dado un nombre (name).

```
public static void deletePilotByName(ObjectContainer db, String
name) {
        ObjectSet result = db.queryByExample(new Pilot(name,
0));
        Pilot found = (Pilot) result.next();
        db.delete(found);
        System.out.println("Deleted " + found);
    }
```

Añadir dentro del bloque `try { }` anterior, la llamada para borrar al piloto “Michael Schumacher” y obtener la información de todos los pilotos.

6. Conexión de la lógica del negocio con el nivel de datos

Implementar la lógica del negocio apropiada para ser utilizada con la presentación realizada en el laboratorio “Separación entre Presentación y Lógica del Negocio"

```
NOTA: LA SOLUCIÓN A ESTE APARTADO Y AL LABORATORIOA
ARQUITECTURAS SOFTWARE SE ENCUENTRA EN EL FICHERO labIso1.zip
```

```
package logic;

public class CheckBD implements Check{

// Definir el código necesario para trabajar con una BD bd4o

 public boolean check(String login, String password){

// Comprueba si existe una cuenta con ese login y password en
// una BD db4o

 }
public boolean add(String login, String password){

// Añade una nueva cuenta a la BD, devolviendo true si no existe
// o false si ya existía


 }

}
```