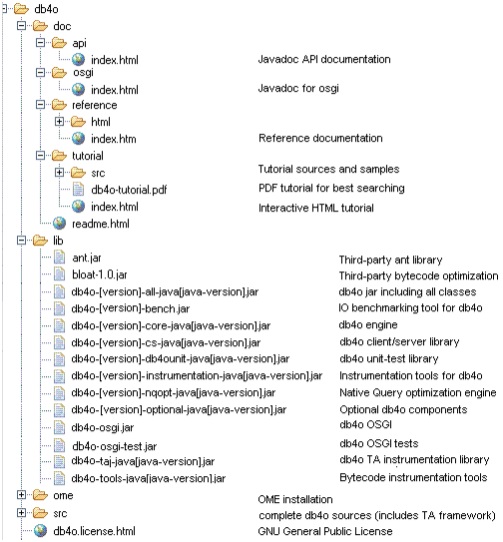
PRACTICE 4-2. DB4O: DB FOR OBJECTS

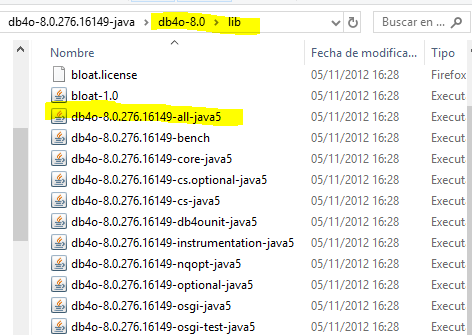
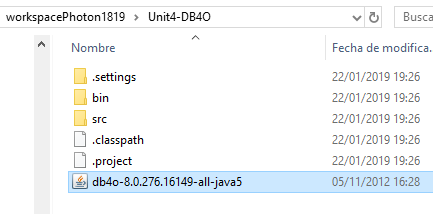
STEP 1. Install db40.

Download it from <https://www.dropbox.com/s/s6farl3dqez7e1r/db4o-8.0.276.16149-java.zip?dl=0>

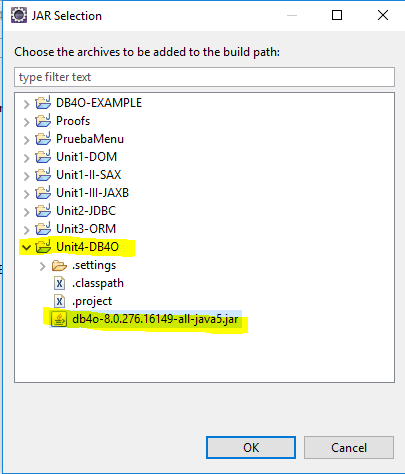
**Unzip it. See the structure of db4o**



**STEP 2: Create a new Project "Unit4-db4o" in Java. Find the file lib/**db4o-XXXX-all-java5.jar and copy it in the root of your Project

Now, we need to add the file in the Build Path>>add JARs…



**Apply and close. You will see it in the Referenced Libraries**

STEP 2. CREATE THE CLASS WITH THE TYPE OF OBJECTS TO STORE IN THE DATABASE.

SEE EXAMPLE IN THE LINK BELOW (GO TO PAGE 8)

<http://www.odbms.org/wp-content/uploads/2013/11/db4o-7.10-tutorial-java.pdf>

First we create a class to hold our data. In our example this will be a Formula One (F1) pilot whose attributes are his name, lastName and the F1 points he has already gained this season. It looks like this:

**public** **class** Pilot {

**private** String name;

**private** String lastName;

**private** **int** points;

**public** Pilot(String name, String lastName, **int** points) {

**this**.name = name;

**this**.lastName = lastName;

**this**.points = points;

}

**public** **int** getPoints() {

**return** points;

}

**public** **void** addPoints(**int** points) {

**this**.points += points;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **void** setLastName(String lastName) {

**this**.lastName = lastName;

}

**public** **void** setPoints(**int** points) {

**this**.points = points;

}

**public** String getName() {

**return** name;

}

**public** String getLastName() {

**return** lastName;

}

**public** String toString() {

**return** name + "-" + lastName + "/" + points;

}

}

COMMENT: The method toString() is a method we can write, and it can be used later to show the information about an object, for example in the instruction System.out.println(myPilot).

STEP 3: CREATE THE CLASS AccessPilotDB4O.java:

**import** com.db4o.Db4oEmbedded;

**import** com.db4o.ObjectContainer;

**import** java.util.List;

**public** **class** AccessPilotDB4O{

**private** ObjectContainer db;

AccessPilotDB4O(){

db = Db4oEmbedded.*openFile*("dbPilotFile.db4o");

}

**void** closeDB() {

db.close();

}

}//class

OPENING THE DATABASE: we need the objects “ObjctContainer” and “D2b4oEmbedded”.

Db4oEmbedded needs the name of the database as an argument(in this case we will call it “dbPilotFile.db4o”

ObjectContainer db = Db4oEmbedded.*openFile*("dbPilotFile.db4o");

//code that Works with the database

db.close()

We must close the database before we leave the application:

We can create a method to do it, if we don’t want to open and close the ObjectContainer in every method:

**void** closeDB() {

db.close();

}

STORING OBJECTS IN THE DATABASE

To store objects, we use the method db.store(object) (db is an ObjectContainer)

**void** insert(String name,String lastName,**int** points) {

Pilot p=**new** Pilot(name,lastName,points);

**try**{

db.store(p);

}**catch**(Exception e) {

System.***out***.println(e);

System.***out***.println("The pilot "+ p+ "could not be inserted");

}

}

We can now create the database and insert pilots with a new class “UsePilotDb40.java”

**public** **class** UsePilotDB4O {

**public** **static** **void** main(String[] args) {

AccessPilotDB4O a=**new** AccessPilotDB4O();

a.insert("pepe","lopez", 100);

a.insert("ana", "garcia", 150);

a.closeDB();

}

}

Insert 5-6 pilots in db4o.

RETREIVING ALL THE OBJECTS IN THE DATABASE:

Let’s do a method to show all the pilots stored in the database: we will use the metod db.query()

**void** showAll() {

**try** {

List<Pilot> pilots = db.query(Pilot.**class**);

**for** (Pilot p : pilots) {

System.***out***.println(p);// we have method toString defined in Pilot

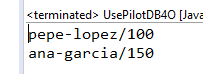
}

}**catch**(Exception e) {

System.***out***.println(e);

}

}



The method query gives an object List, and we can traverse the list with the for(Pilot p: pilots) (available from Java 5)

ObjectContainers have also the method queryByExample().The result of the query can also be kept in a variable ObjectSet. This variable has the methods size(), hasNext() and next(). We

**void** showAll2() {

Pilot p = **new** Pilot(**null**,**null**, 0);

ObjectSet result = db.queryByExample(p);

System.***out***.println(result.size());

**while**(result.hasNext()) {

System.***out***.println(result.next());

}

}

COMMENT: In the line Pilot p = **new** Pilot(**null**,**null**, 0);

Write the following p = **new** Pilot(**“ana”**,**null**, 0);

QUESTION: What result do you get with that code?

UPDATING OBJECTS: WE find the object, change the properties and store it again.

**void** updatePoints(String name,String lastName, **int** newPointsToAdd) {

ObjectSet result = db.queryByExample(**new** Pilot(name,lastName, 0));

Pilot found = (Pilot) result.next();

found.addPoints(newPointsToAdd);

db.store(found);

System.***out***.println("Added points for " + found);

showAll();

}

Notice that we query for the object first. This is an importaint point. When you call #store() to modify a stored object, if the object is not 'known' (having been previously stored or retrieved during the current session), db4o will insert a new object. db4o does this because it does not automatically match up objects to be stored, with objects previously stored. It assumes you are inserting a second object.

DELETING OBJECTS:

**void** delete(String name, String lastName) {//delete the first object found

ObjectSet result = db.queryByExample(**new** Pilot(name,lastName, 0));

Pilot found = (Pilot) result.next();

db.delete(found);

System.***out***.println("Deleted " + found);

}

* Delete one or two of the objects inserted and see that the method works.