PRACTICE 6.2. JavaBeans to Access DataBases.

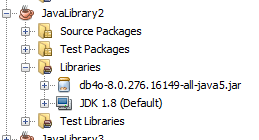
OBJECTIVE: We will create a JavaBean to access a database db4o.

## REALIZATION:

IMPORTANT COMMENT: Work in your JavaLibrary1 created in Practice6.1. In the images it appears “JavaLibrary2”, but you HAVE TO WORK WITH JavaLibrary1 project

STEP1: **Find the file lib/**db4o-XXXX-all-java5.jar and copy it in the root of your Project JavaLibrary1.

Right click on the folder “Libraries” of your project and select “Add JAR/folder”. Select the db40 jar.



STEP 2: Create the following classes:

Class Sale:

First define the properties, second create the constructor, finally create the getters and setters.

**package** MyBeans;

**import** java.beans.\*;

**import** java.io.Serializable;

**import** java.util.Date;

**import** java.beans.PropertyChangeEvent;

**import** java.beans.PropertyChangeListener;

**public** **class** Sale **implements** Serializable, PropertyChangeListener {

**private** **int** saleNumber;

**private** **int** productId;

**private** Date date;

**private** **int** amount;

**private** String comment;

**public** Sale() {

}

**public** Sale(**int** saleNumber, **int** productId, Date date, **int** amount, String comment) {

**this**.saleNumber = saleNumber;

**this**.productId = productId;

**this**.date = date;

**this**.amount = amount;

**this**.comment = comment;

}

**public** **void** propertyChange(PropertyChangeEvent evt){

**this**.comment="Waiting order for lack of stock";

}

**public** **int** getSaleNumber() {

**return** saleNumber;

}

**public** **void** setSaleNumber(**int** saleNumber) {

**this**.saleNumber = saleNumber;

}

**public** **int** getProductId() {

**return** productId;

}

**public** **void** setProductId(**int** productId) {

**this**.productId = productId;

}

**public** Date getDate() {

**return** date;

}

**public** **void** setDate(Date date) {

**this**.date = date;

}

**public** **int** getAmount() {

**return** amount;

}

**public** **void** setAmount(**int** amount) {

**this**.amount = amount;

}

**public** String getComment() {

**return** comment;

}

**public** **void** setComment(String comment) {

**this**.comment = comment;

}

}

STEP: CREATE CLASS DataBase to have all the methods for the access to db4o.

**package** MyBeans;

**import** java.beans.\*;

**import** java.io.Serializable;

**import** java.util.List;

**import** com.db4o.Db4oEmbedded;

**import** com.db4o.ObjectContainer;

/\*\*

\*

\* **@author** ines

\*/

**public** **class** DataBase **implements** Serializable {

**static** ObjectContainer *db*;

**public** DataBase(){

//open the database

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

}

**public** **static** ObjectContainer getOC(){

**return** DataBase.*db*;

}

**public** **static** **int** getOrderNumber(){

**int** maxOrder=0;

List<Order> orders = *db*.query(Order.**class**);

**for** (Order order : orders) {

**if** (order.getOrderNumber()>maxOrder){

maxOrder=order.getOrderNumber();

}// we have method toString defined in Pilot

}

**return** maxOrder+1;

}

**public** **int** getSaleNumber(){

**int** maxSale=0;

List<Sale> sales = *db*.query(Sale.**class**);

**for** (Sale sale : sales) {

**if** (sale.getSaleNumber()>maxSale){

maxSale=sale.getSaleNumber();

}// we have method toString defined in Pilot

}

**return** maxSale+1;

}

//insert a product

**public** **void** insertProduct(Product p){

*db*.store(p);

*db*.commit();

}

**public** **static** **void** insertOrder(Product p, **int** amount){

Order order=**new** Order(*getOrderNumber*(),p,*getCurrentDate*(),amount);

*db*.store(order);

System.***out***.printf("ORDER GENERATED for the product: %s%n",p.getDescription());

*db*.commit();

}

**public** **void** insertSale(Product p, **int** amount){

**int** saleNumber=getSaleNumber();

Sale sale=**new** Sale(saleNumber,p.getIdProduct(),*getCurrentDate*(), amount, " ");

p.addPropertyChangeListener(sale);//adding a listener

updateStock(p,amount);

*db*.store(sale);

System.***out***.printf("SALE %d INSERTED, Comments: %s %n", saleNumber, sale.getComment());

*db*.commit();

}

**private** **void** updateStock(Product p, **int** amount){

Order order=**new** Order();

order.setProduct(p);

order.setAmount(amount);

p.addPropertyChangeListener(order);//adding a listener

//calculate the stock

**int** newStock=p.getStock()-amount;

p.setStock(newStock);

*db*.store(p);

*db*.commit();

}

**public** **static** **void** closeDB() {

*db*.close();

}

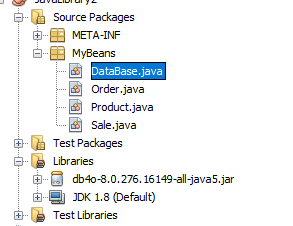
**private** **static** java.sql.Date getCurrentDate(){

java.util.Date today=**new** java.util.Date();

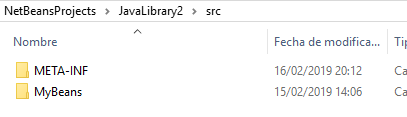
**return** **new** java.sql.Date(today.getTime());

}

}



STEP: CREATE THE FOLDER META-INF WITH THE FILE MANIFEST.MF:You need to create the folder yourself META-INF and the file MANIFEST.MF with “bloc de notas”



Change the content of MANIFEST\_MF:

Manifest-Version: 1.0

Name: MyBeans/Product.class

Java-Bean: True

Name: MyBeans/Order.class

Java-Bean: True

Name: MyBeans/Sale.class

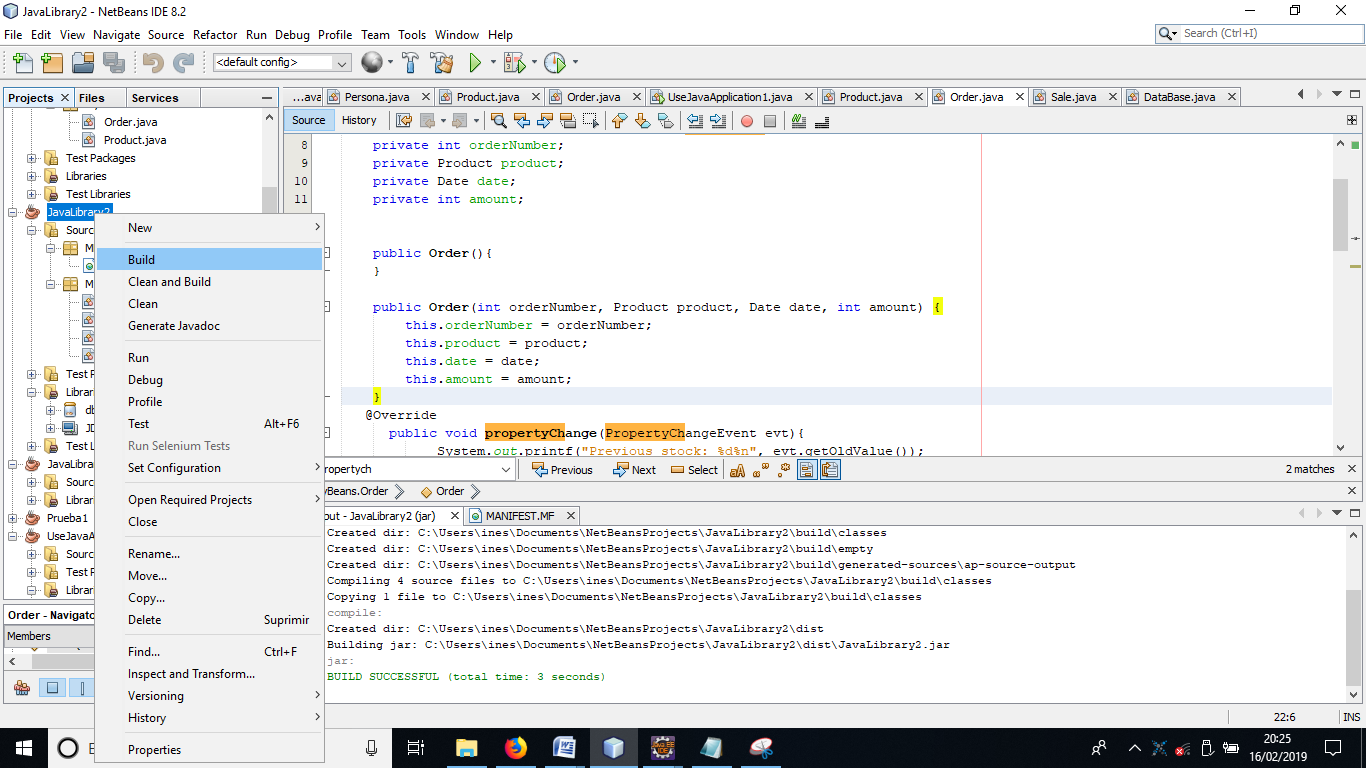
Java-Bean: True

Name: MyBeans/DataBase.class

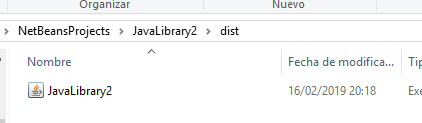
Java-Bean: False

(REMEMBER TO INSERT A LINE AT THE END OF THE FILE)

Now, build the project:



The file .JAR is in the folder dist of the project:



STEP3 : USE the jar file. Create a new Project ProdOrderSale. Add the Libraries jar created. Create a new class to try the code. Its name is “FillProducts” to insert products in our database.

New>>File>>filter class>>Java Main class.

**package** product.order.sale;

**import** MyBeans.DataBase;

**import** MyBeans.Product;

**public** **class** FillProducts {

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

DataBase db=**new** DataBase();

Product p1=**new** Product("DDR memory",1,10,3,50);

Product p2=**new** Product("monitor 15i",2,5,2,176);

Product p3=**new** Product("ssd",3,20,5,65);

Product p4=**new** Product("monitos 17i",4,6,2,200);

//store the products

db.insertProduct(p1);

db.insertProduct(p2);

db.insertProduct(p3);

db.insertProduct(p4);

//see the products

db.closeDB();

}

}

RUN THE FILE ONLY ONE TIME, TO INSERT PRODUCTS IN YOUR DATABASE.

STEP 4: Create a new executable class “ShowProducts” to access the products in the database and see that it works:

**package** product.order.sale;

**import** MyBeans.DataBase;

**import** MyBeans.Order;

**import** MyBeans.Product;

**import** MyBeans.Sale;

**import** java.util.List;

**import** com.db4o.Db4oEmbedded;

**import** com.db4o.ObjectContainer;

**public** **class** ShowProducts {

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

System.***out***.println("Connecting to dbSales.db4o");

DataBase db=**new** DataBase();

ObjectContainer oc=DataBase.getOC();

System.***out***.println("----------PRODUCTS LIST------------");

List<Product> products = oc.query(Product.**class**);

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

**for** (Product p : products) {

System.***out***.println(p.getIdProduct()+"-"+p.getDescription()+"STOCk:"+p.getStock()+"-"+p.getPvp());

}

System.***out***.println("----------SALES LIST------------");

List<Sale> sales=oc.query(Sale.**class**);

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

**for** (Sale s : sales) {

System.***out***.println(s.getSaleNumber()+"-"+s.getProductId()+"-"+s.getDate()+"-"+s.getAmount());

}

System.***out***.println("----------ORDERS LIST------------");

List<Order> orders=oc.query(Order.**class**);

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

**for** (Order o : orders) {

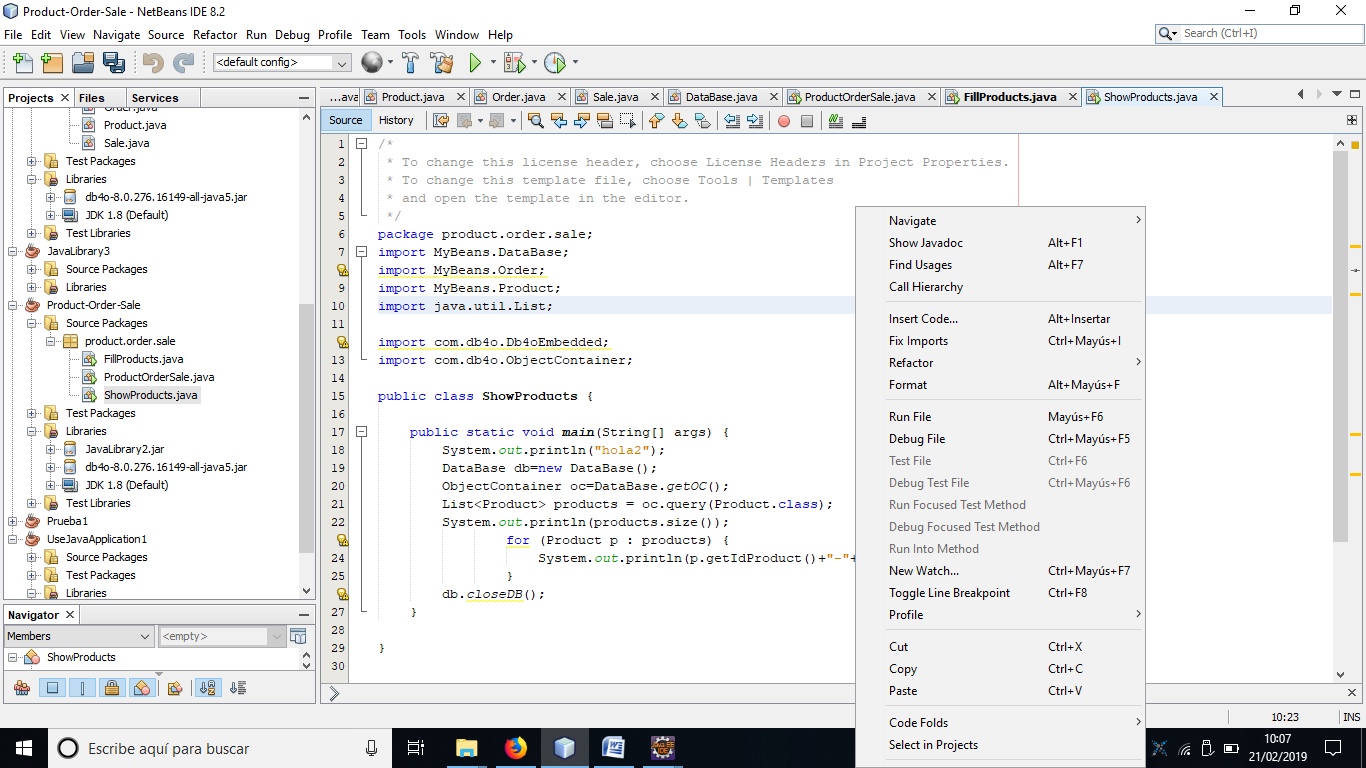
System.***out***.println(o.getProduct().getDescription()+"-"+o.getOrderNumber()+"-"+o.getDate()+"-"+o.getAmount());

}

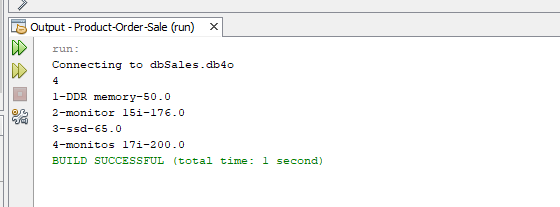
db.closeDB();

}

}



To run the code, right-click on the code of the executable file and select “Run File”



STEP 5: Let’s do some code to create sales, and see if orders are generated automatically if the stock gets less than minimun. Create and executable class ProductOrderSale with the following code:

**import** MyBeans.DataBase;

**import** MyBeans.Product;

**import** MyBeans.Sale;

**import** com.db4o.ObjectContainer;

**import** com.db4o.ObjectSet;

**import** java.util.List;

**public** **class** ProductOrderSale {

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

// **TODO** code application logic here

System.***out***.println("Accessing the database dbSales to insert one sale of a product----");

DataBase db=**new** DataBase();

ObjectContainer oc=DataBase.getOC();

**int** idProduct=1;//DDR memory, stock 10, minimunStock 3

**int** amount=8;//if we sell 8, let's see if an order is generated

//public Sale(int saleNumber, int productId, Date date, int amount, String comment) {

Product p=**new** Product(**null**,idProduct,0,0,0);

//voy por aquí

Product nextProd=**null**;

ObjectSet<Product> result = oc.queryByExample(p);//obtiene el producto con la idProduct

//comprobamos que encuentra el producto con id=1, que es la DDR memory

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

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

nextProd=result.next();

System.***out***.println(nextProd.getDescription()+"-"+nextProd.getStock());

}

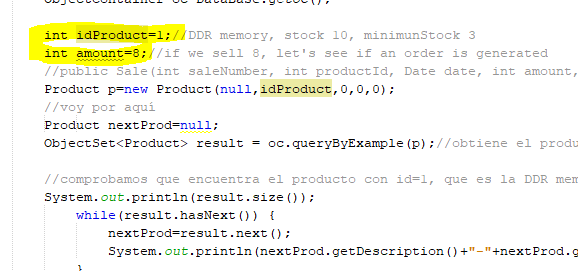
//creamos una venta del producto. Si el stock decrece, debe generarse un pedido

db.closeDB();

}

}

DO SEVERAL SALES CHANGING THE VALUES OF idProduct AND amount.



FINALLY, run the class “Show Products” to see the information about products (check that the stock has changed), sales and orders:

