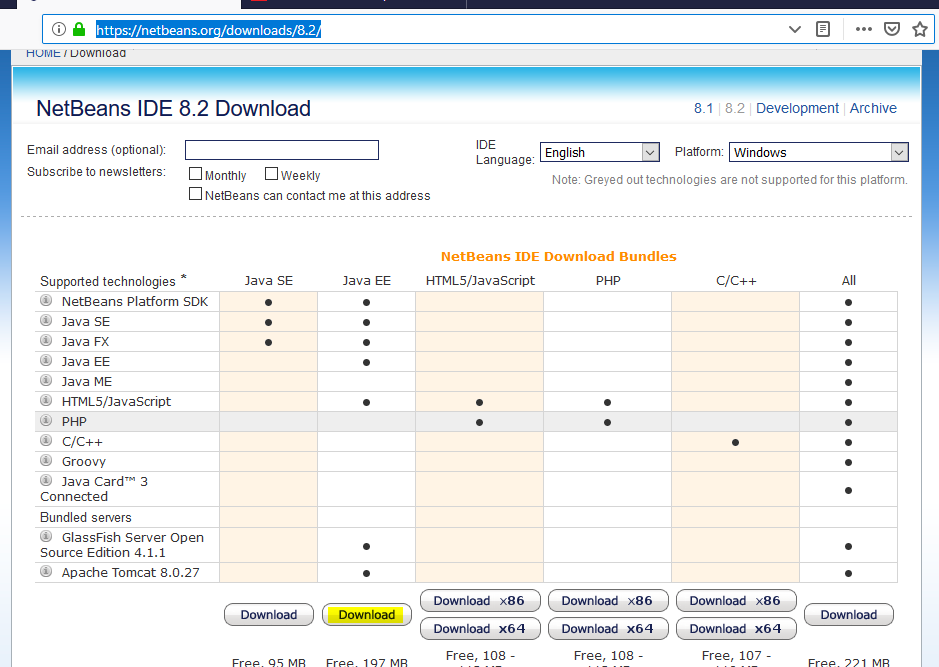
PRACTICE 6.1: CREATING COMPONENTS WITH NETBEANS

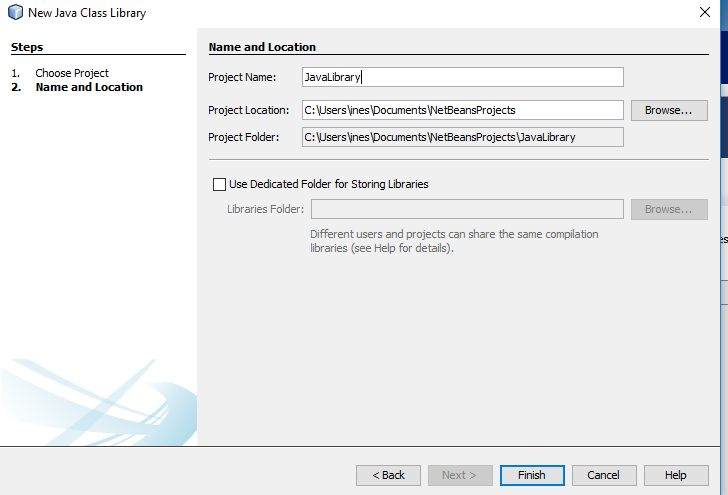
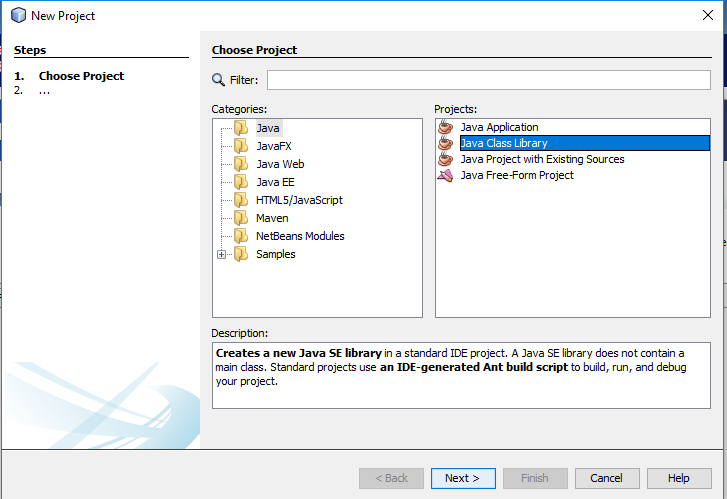
STEP1: Download Neatbeans 8.2 and install it.

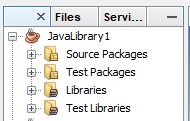
<https://netbeans.org/downloads/8.2/>



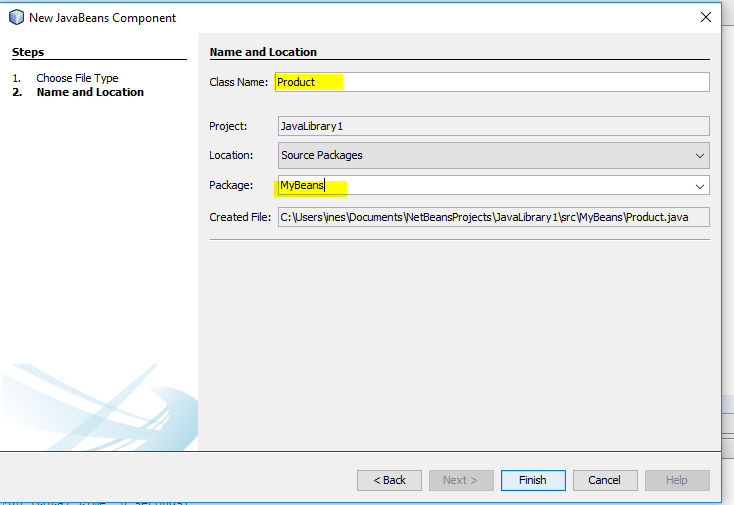
STEP2: Open Netbeans. We want to create our own library with the Beans “Product” and “Order”.

First of all, we create a new Java Class Library ***JavaLibrary1***: File>>New Project>>Java Class Library

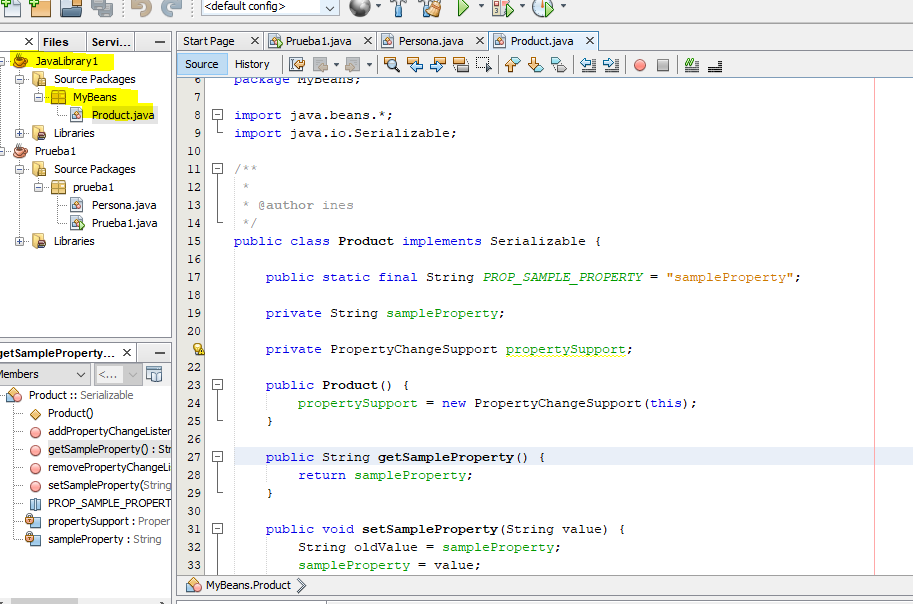




Now we can create the beans inside that project. Let’s do the bean “Product” first:Rigth click on project JavaLibrary1>>New>>Other>>JavaBeans Objects>>JavaBean Component.



When we accept, the following code is generated automatically:



We change the code generated: delete sampleProperty and metods getSampleProperty(), setSampleProperty. Include the following:

**public** **static** **final** String ***PROP\_SAMPLE\_PROPERTY*** = "sampleProperty";

**private** String description;

**private** **int** idProduct;

**private** **int** stock;

**private** **int** minimunStock;

**private** **double** pvp;

**private** PropertyChangeSupport propertySupport;

**public** Product(String description,**int** idProduct, **int** stock, **int** minimunStock, **double** pvp) {

propertySupport=**new** PropertyChangeSupport(**this**);

**this**.description = description;

**this**.idProduct = idProduct;

**this**.stock = stock;

**this**.minimunStock = minimunStock;

**this**.pvp = pvp;

}

Generate the getters and setters. Right click under the constructor (where you will insert the code) and choose Insert Code>>Getter and Setter…

Then choose all the properties except propertySupport and Generate.

Change the method “setStock” to generate an event if the stock is smaller than the minimum required(with the method) firePropertyChange(), to leave the stock with the old value

**public** **void** setStock(**int** stock) {//modify this method

**int** oldStock=**this**.stock;//save the current value

**this**.stock = stock;

**if**(**this**.stock<getMinimunStock()){//we need to make an order (order=pedido)

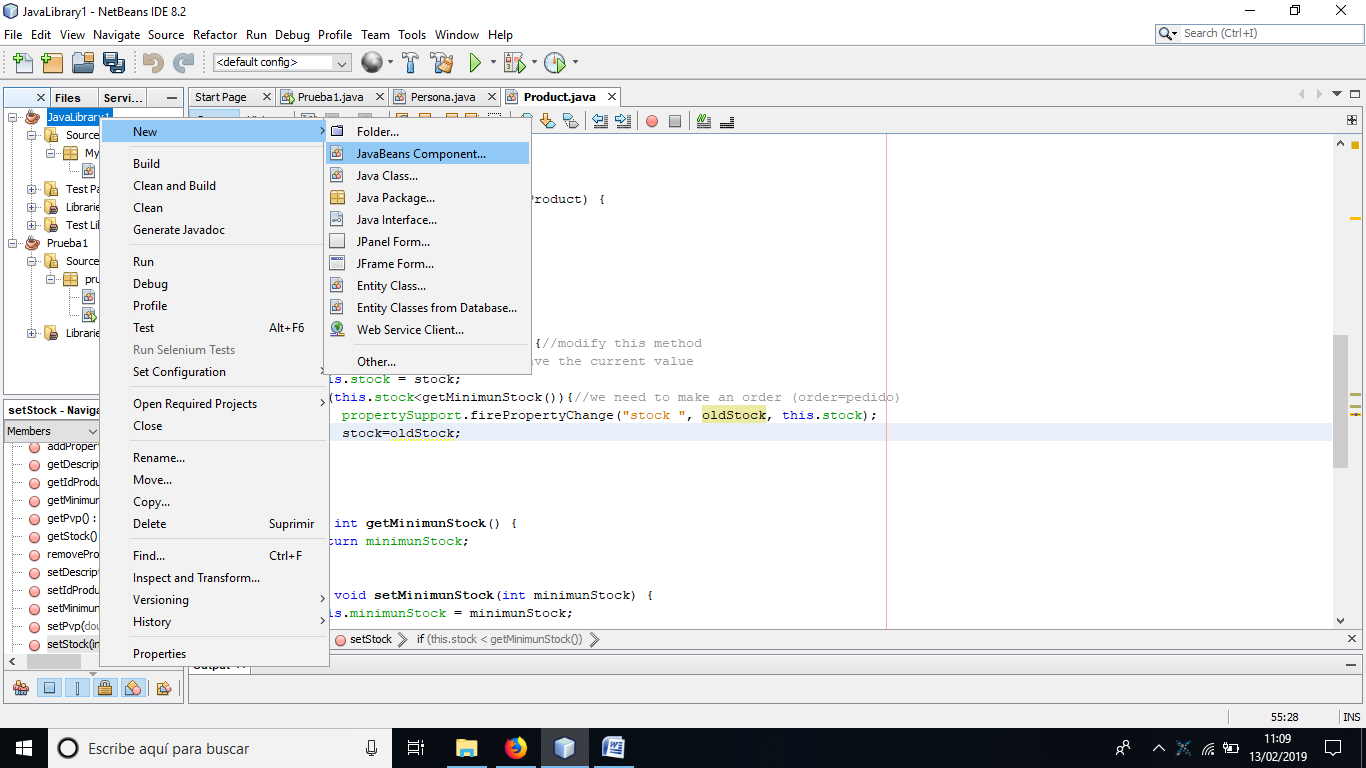
propertySupport.firePropertyChange("stock ", oldStock, **this**.stock);

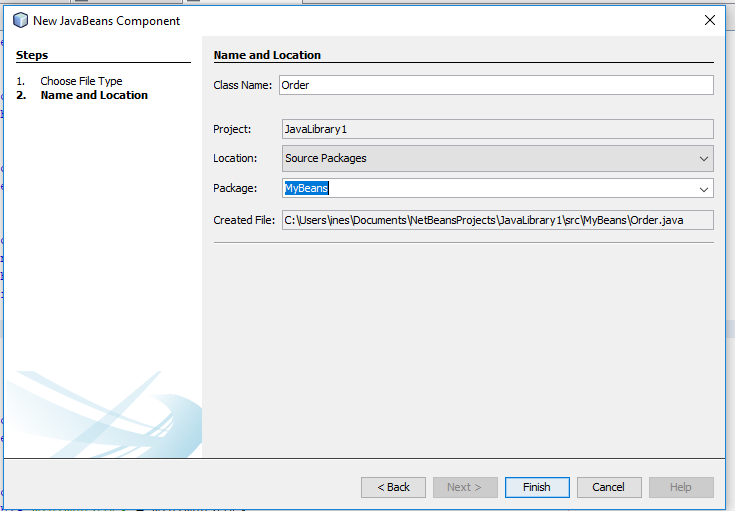
stock=oldStock;

}

}

STEP 3: Create another JavaBean for “Order” in the package “MyBeans”.





Delete all the code of the class.

Add **PropertyChangeListener** to the sentence implements.

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

We have to overwrite the method **propertyChange()**: Write the code:

@Override

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

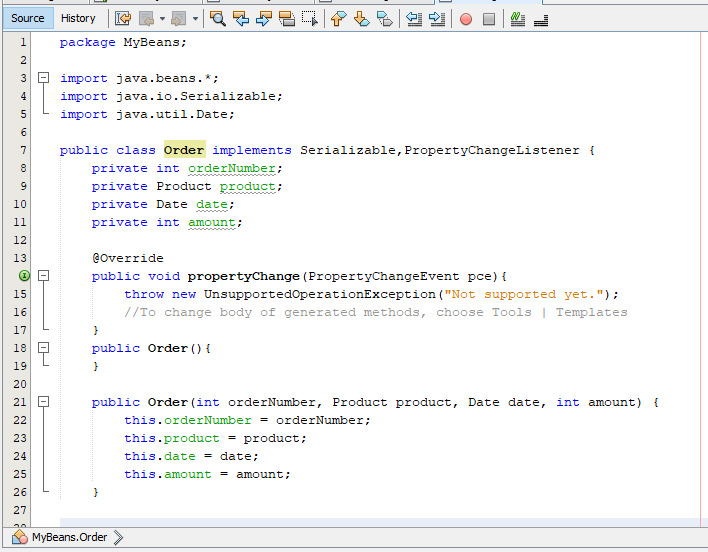
System.***out***.printf("Old stock: %d%n", evt.getOldValue());

System.***out***.printf("Stock: %d%n", evt.getNewValue());

System.out.printf(("Create an order for product : %s%n", product.getDescription());

}

Add the properties, create the constructors:



Insert the getters and setters for the properties: Source>>Insert Code..>>Getter and Setter…

The final code must be this:

**package** MyBeans;

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

**import** java.io.Serializable;

**import** java.util.Date;

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

**private** **int** orderNumber;

**private** Product product;

**private** Date date;

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

@Override

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

System.***out***.printf("Old stock: %d%n", evt.getOldValue());

System.***out***.printf("Stock: %d%n", evt.getNewValue());

System.out.printf(("Create an order for product : %s%n", product.getDescription());

}

**public** Order(){

}

**public** Order(**int** orderNumber, Product product, Date date, **int** amount) {

**this**.orderNumber = orderNumber;

**this**.product = product;

**this**.date = date;

**this**.amount = amount;

}

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

**return** orderNumber;

}

**public** **void** setOrderNumber(**int** orderNumber) {

**this**.orderNumber = orderNumber;

}

**public** Product getProduct() {

**return** product;

}

**public** **void** setProduct(Product product) {

**this**.product = product;

}

**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;

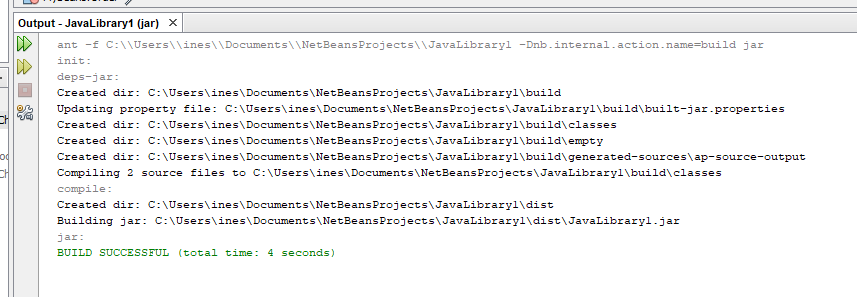
}

}

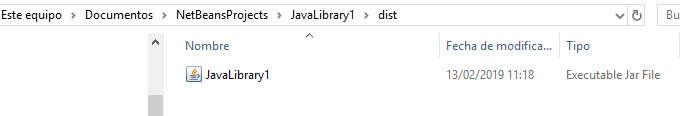
STEP 4: Now that we have the JavaBeans, let’s generate the JAR.

Right click on the project>>Build

You’ll see the messages in the Output window about the generation:

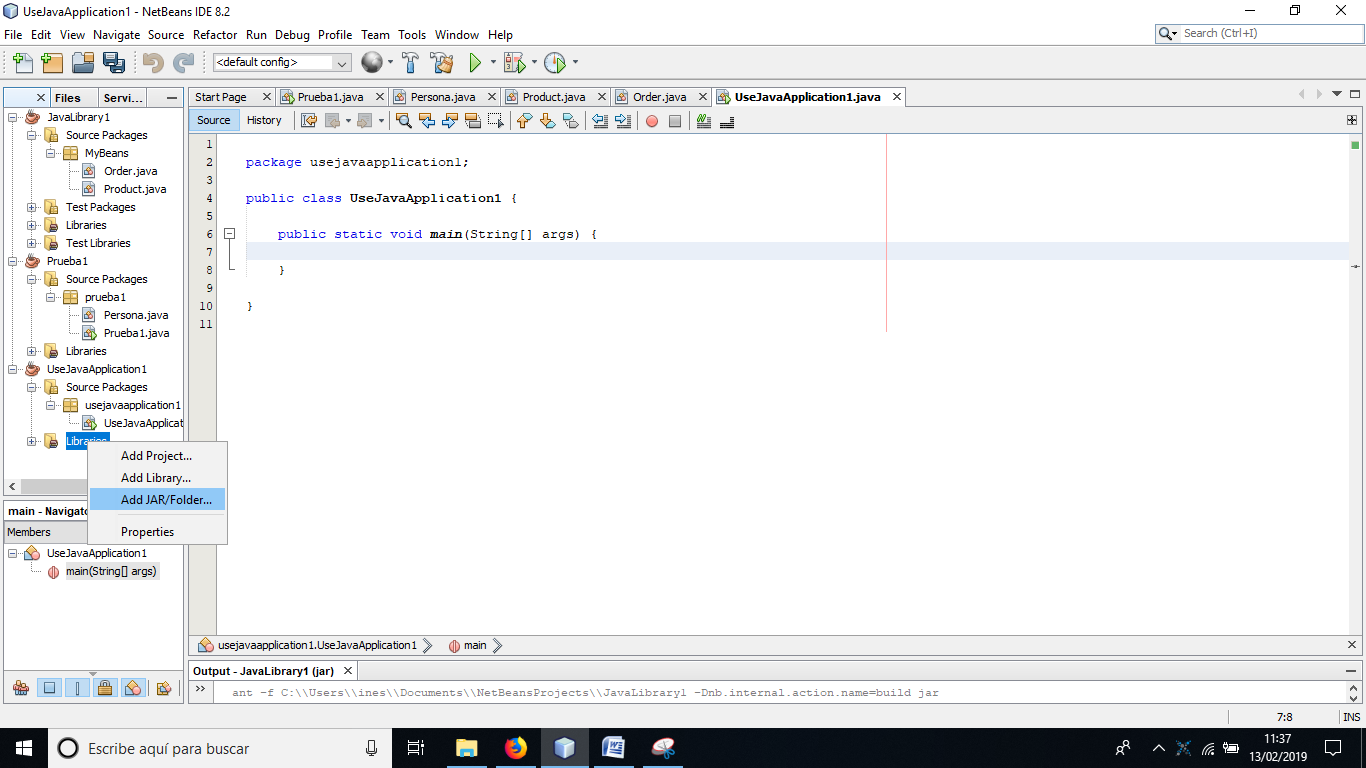
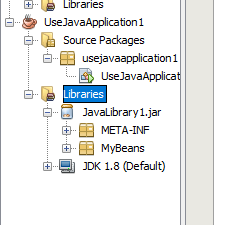


If it is OK, you’ll hate the file ***LibraryJava1.jar*** in the directory ***dist*** of the project:

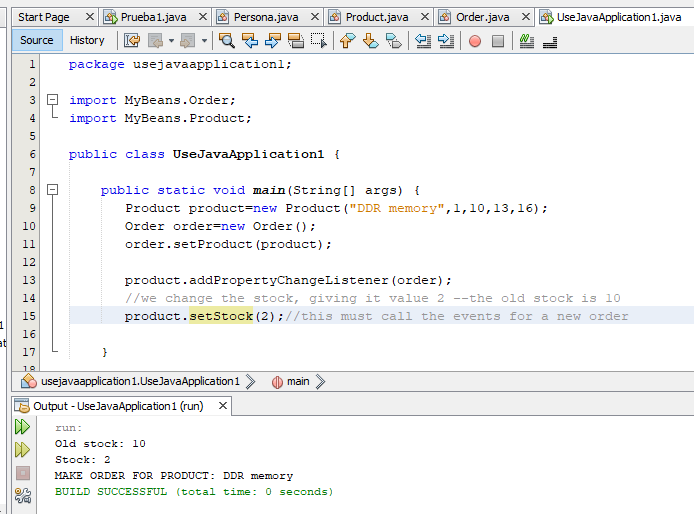


If we modify the JavaBeans, we must select the option Clean and Build the library again:

STEP 5: To try our components, let’s create a new project: UseLibraryJava1. Include the LibraryJava1.jar (Librares>>Add JAR Folder)

STEP : Write a code to try the library:

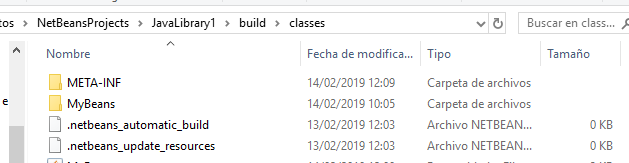


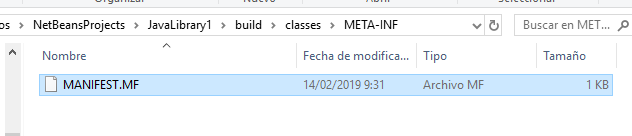
STEP : CREATE A PACKAGE WITH THE COMPONENTS:

To distribute the jar file, we must have a file MANIFEST.MF describing the content of the JAR.

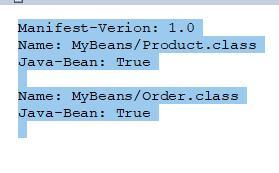
<http://gpd.sip.ucm.es/rafa/docencia/programacion/tema1/jar.html>

We need the following structure:



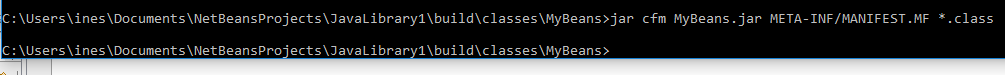


The content of MANIFEST.MF is the following: (It needs a while line at the end of the file)



Open the cmd and go to ..\NetBeansProjects\JavaLibrary1\build\classes>

jar cfm MyBeans.jar META-INF\MANIFEST.MF MyBeans\\*.class



Now you have your component created: MyBeans.jar.

