An executable JAR file

A JAR (Java ARchive) is a way of packing together all resources associated with a program (compiled classes, source files, images, sounds, etc). Putting your program in a JAR allows it to be distributed as a single executable file, saving space and simplifying the download process. It also runs consistently in the same way.

This tutorial will help you understand how to build such executable programs from the command line.

A simple example - Hello World

Let's say we wanted to distribute the canonical program *"Hello, world!"*.

Start by opening the terminal/command prompt.

Classes in java should be created inside packages. Each package in java MUST coincide with a folder/directory in your file system. So, create a package named hello, that is, create a *hello* folder. Inside the *hello* folder, create a text file named *HelloWorld.java* with the following content:

```
package hello;
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

Next, being in the terminal at the level of the *hello* folder, that is, when you type *ls*/*dir* you see the directory *hello,* compile the *.java* file by doing this:

```
javac hello/*.java
```

This command compiles the *HelloWorld.java*. The * represents the wildcard and matches any file name inside the *hello* folder with extension *.java*. Check that inside the package *hello* now you also have the compiled class *HelloWorld.class*.

Next, again at the level of the *hello* folder, create a text file named *manif.txt* which contains:

```
Main-Class: hello.HelloWorld
```

This file is called the MANIFEST and it should have no other character/text, it is space- and case-sensitive, and it MUST end with a blank line. You really need to respect uppercase/lowercase letters, spaces, and the blank line in the final!

This file says that the class with the main you want this executable to run is inside the package *hello* within the compiled class *HelloWorld.class*.

Then, create the JAR file by typing:

```
jar cmf manif.txt HelloWorld.jar hello/*
```

The * is a wildcard that tells the terminal to include everything that is inside the *hello* package. In this example, it will be the *HelloWorld.java* and the *HelloWorld.class*. If you do not want to include the source code (your *.java files) in the JAR use only: hello/*.class (instead of hello/*).

Finally, run your executable JAR file by typing:
java -jar HelloWorld.jar

You can check whatever you have inside the JAR archive by typing:
jar tf HelloWord.jar
which should print to the terminal the following:
META-INF/
META-INF/MANIFEST.MF
hello/HelloWorld.java
hello/HelloWorld.class

Note: the contents of your *manif.txt* were copied to the MANIFEST.MF file and placed inside the META-INF folder by the JAR tool.

The JAR file *HelloWorld.jar* can now be downloaded and executed.

Creating an executable JAR file

Here is the general procedure for creating an executable JAR:

1. Compile your java code, generating all of the program's class files:
   javac *input-files*
   If you have more than one package just list all packages/files in the *input-files*, e.g.:
   pck1/*.java pck2/*.java pck3/subpck31/*.java pck3/subpck32/*.java

2. Create a *manifest file* containing the following line:
   Main-Class: *package path to the class containing the main (blank line)*

3. To create the JAR, type the following command:
   jar cmf *manifest-file jar-file input-files*
   The *input-files* must include any class files, images, sounds, etc, that your program uses. Optionally, you can include the program's .java files in the JAR.
   If you have more than one package just list all packages/files in the *input-files*, e.g.:
   pck1/* pack2/* pck3/subpck31/* pck3/subpck32/*

4. To view the contents of the JAR, type:
   jar tf *jar-file*

5. Execute the application from the command line by typing:
   java -jar *jar-file*