Math 4610 Lecture Notes

How to Build a Jar File for Java Objects *

Joe Koebbe

October 6, 2019

*These notes are part of an Open Resource Educational project sponsored by Utah State University

Steps for Building a Shared Library:

Like shared libraries that contain some number of object files, a jar file can be used to collect class files that are generated by compiling java code. This set of notes will cover how to create jar files that collect class files. In addition, we will see how to extract and use a subset of the class files for applications and to attach to other java codes.

Setting Things Up

To start, let's create a couple of java files that define simple objects that return the absolute error and the relative error. It should be noted that this may not be the best way to define these routines. However, we are after a simple example so that the details of the jar file creation are not lost in the work in this example.

The Codes We Have:

The following code can be edited into a text file. The code below returns the absolute error in using one number to approximate the other.

```
//
// java object that computes and returns the absolute error in approximating
// a real number, y, by another real number x
//
public class AbsError extends Object {
  public static double abserr(double x, double y) {
    return Math.abs(x-y);
  }
}
```

A second ocde for the relative error can be written that looks like the following.

```
//
// define the class for the absolute error object
// ----------------------------------------------
//
  public class RelError extends Object {
    public static double relerr(double x, double y) {
    return Math.abs((x-y)/x);
  }
}
```

So, let's create a folder under the main folder, say src. Do this in a convenient as follows

```
mkdir java
cd java
```

Create the files in the folder. You can either cut and paste the lines or type them in. Next, compile the two files in the folder as follows:

```
% javac *.java
```

Note that the asterisk is a wildcard character which allows the command to compile both files with a single command. In the setting of creating a location for class files, there is one more file that is needed. The jar utility requires the specification of an entry point into any jar file. The entry point can be included in a manifest file that can be created by hand.

For this example, we will create another java class that acts as a default entry point for the library of classes in the jar file. So, create a new file named ObjectInventroy.java using a text editor. Edit or copy in the lines below.

```
//
// define the class for the absolute error object
// ----------------------------------------------
//
import java.io.*;
public class ObjectInventory extends Object {
  public static void main(String args[]) {
    String s = "Use:\n\n"
             + "    jar tf\n\n"
             + "to see a list of the objects in the jar file. Use:\n\n"
             + "    jar xf \n\n"
             + "to extract the objects needed.\n\n";
    System.out.println(s);
  }
}
```

You will need to compile this file as part of the process. So, type the command

```
% javac ObjectInventory.java
```

This will place another class file in the folder. If you look at the object, all it does is print out a message that documents how to use the jar file command to access and extract class files.

The last thing to do is to define the entry point in a manifest file. So, create/edit the manifest file for the jar file. The file must be named mainClass in order to work. So, creat the file using

```
% vim ObjectInventory.java
```

and make the file look like the following.

```
Main-Class: ObjectInventory
```

Save this file and exit your text editor.

The final step is to create the jar file this is done using the following command:

```
% jar cfm MyJar.jar mainClass *.class
```

The flags in the command are defined below:

```
c - tells jar to create the jar file (named MyJar.jar above)
f - the file name for the jar file (MyJar.jar)
m - use the manifest or entry point defined in the mainClass file
```

The last token in the command is the *.class which tells the jar command to place all class files in the current folder in the jar file. To see what is in a jar file use the following command:

```
jar tf MyJar.jar
```

In the simple example here the output looks like:

```
META-INF/
META-INF/MANIFEST.MF
AbsError.class
ObjectInventory.class
RelError.class
```

To execute a program inside a jar file, one specifies this using the following.

```
% java -jar ./MyJar.jar
```

The output from this command for this example is

```
Use:

    jar tf

to see a list of the objects in the jar file. Use:

    jar xf [object.class]

to extract the objects needed.
```

In this example, there will be four files:

- old_main.f - all of the code is in one file
- main.f - just the main part of the code
- smaceps.f - this file contains the code to compute the machine epsilon
- abserr.f - the routine that computes the absolute error
