Nick Troccoli
CS 106A

Handout #5
August 9, 2017

Packaging Your Program into a Distributable JAR File

Based on a handout by Eric Roberts and Brandon Burr

Now that you've written all these wonderful programs, wouldn't it be great if you could package them up and send them to your friends and family so that they could see what you've done? JAR files, or "Java ARchive" files, let you do just that. Here's a short guide to making an executable JAR file in Eclipse!

Step 1: Adding a `main` method

Our programs that have used the ACM libraries and CS 106A Eclipse have started running via the **`public void run()`** method. We changed Eclipse to allow this, to make things easier for you. But in reality, a Java program needs to start at a particular method in a class, the **`public static void main(String[] args)`** method. In order to export your program you'll need to edit your code to explicitly have **`main()`**. Then your program should run fine in any Java compiler. You can do this by adding the following code in the class that has **`public void run()`**, substituting the name of that class for **`'MyClass'`** below.

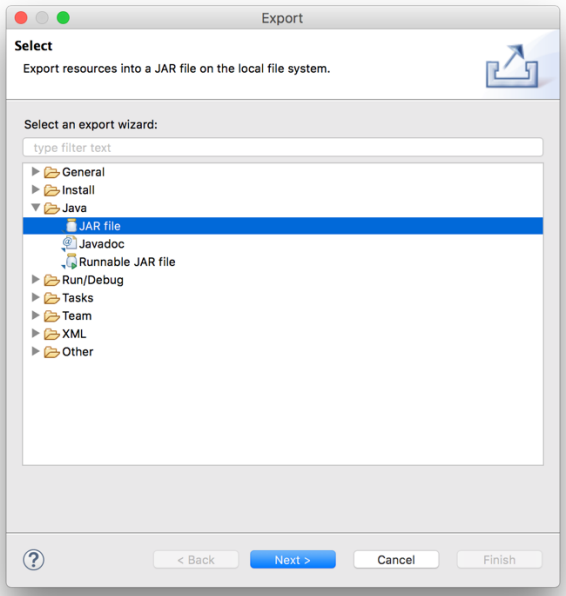
```
public static void main(String[] args) {
      (new MyClass()).start(args);
}
```

Step 2: Exporting a JAR File

Now that we have a normal running Java program, let's package it up into a JAR file. A JAR file is simple a *J*ava *AR*chive – a file that contains a set of Java class files as well as potentially other files that will be used by the program. As part of this, we need to export both a JAR file *and* a **manifest file**. The manifest file allows you to specify things like which is the **`main`** class to run when the JAR file is double-clicked, or what the external libraries are, or even security information about the JAR file. (If you aren't using other JAR files, you don't need to use the manifest file.)

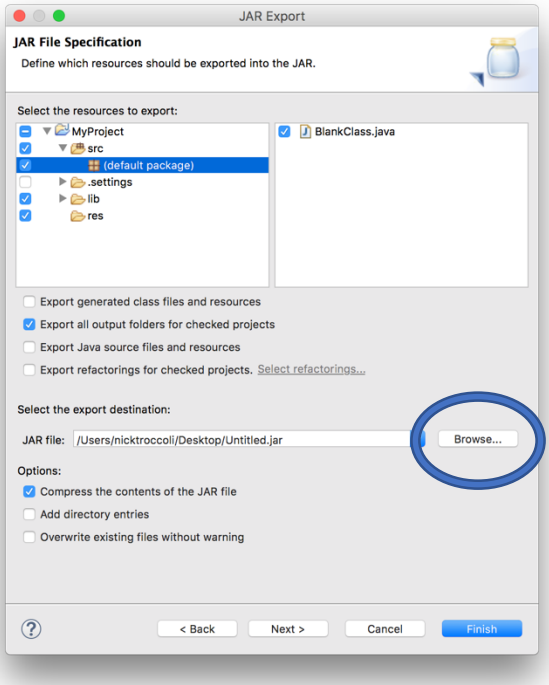
Follow the steps on the following pages to export your program.

1. Click the project folder in the Eclipse sidebar that you want to export, and click **`File -> Export…`** The following window should appear:



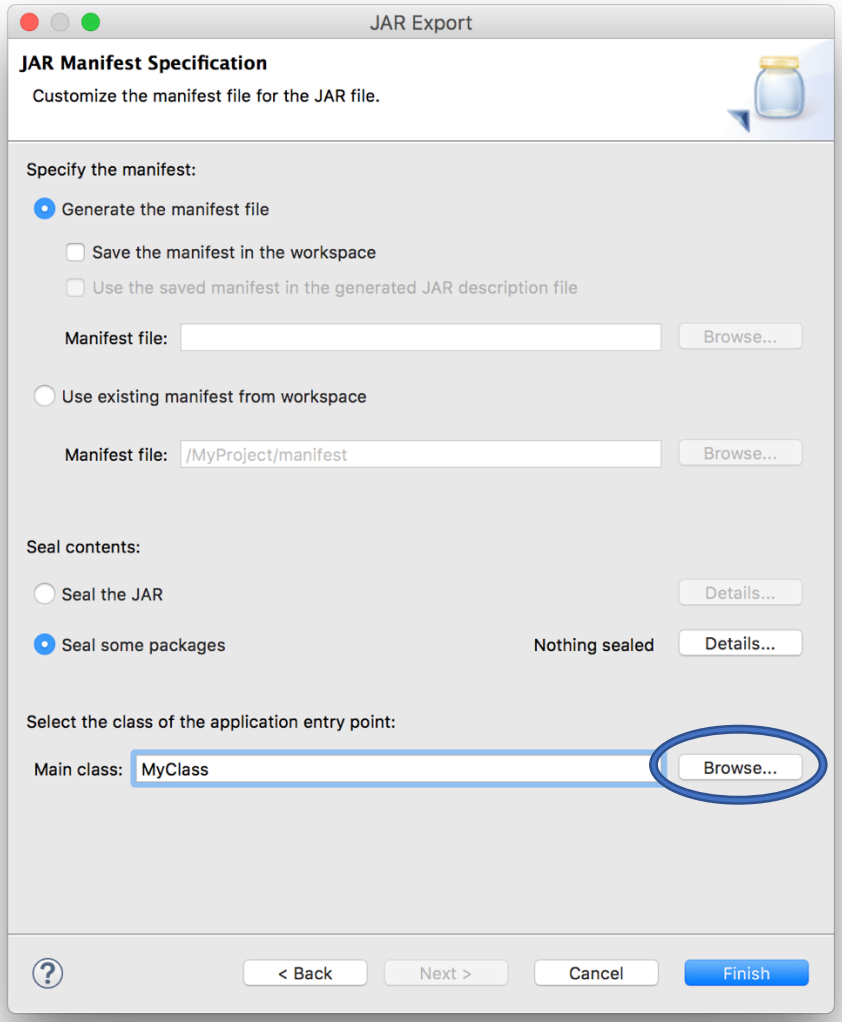
2. Expand the **`Java`** folder, and select **`JAR File`**. Then click **`Next`**.

3. You should see the JAR export window. Expand the project folder on the left and make sure that the **`default package`** is selected, and select the destination of where you want to save the JAR file using the **`'Browse..."`** button. Then hit **`Next`**.



4. Click **`Next`** again on the following screen.

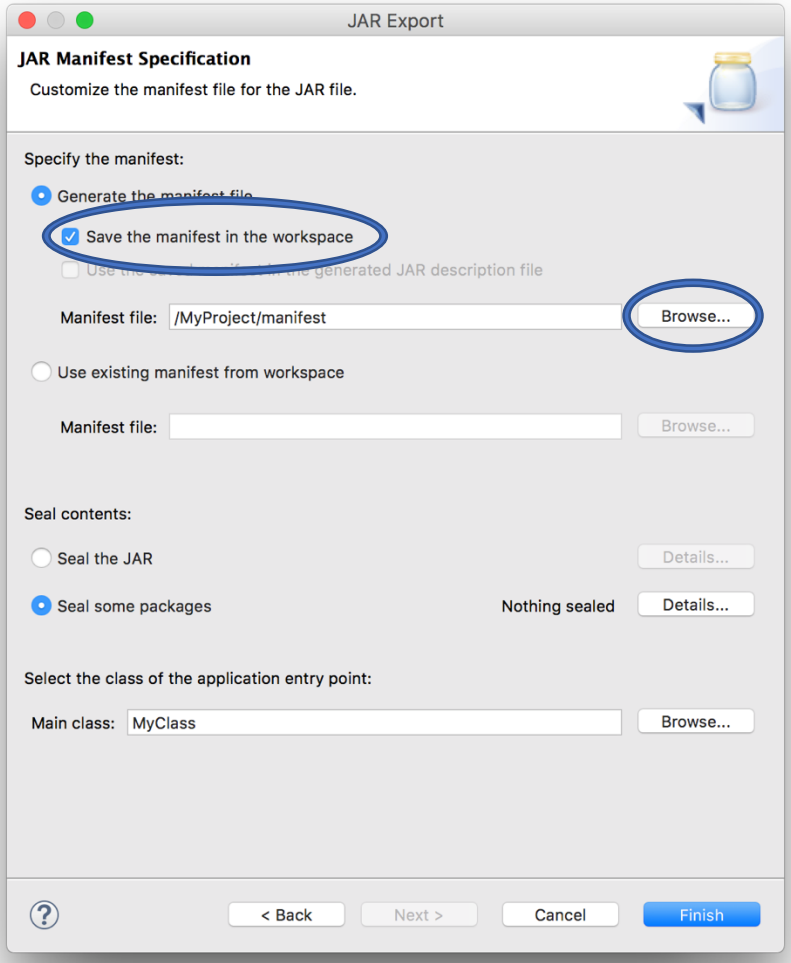
5. Now you will come to the final screen:



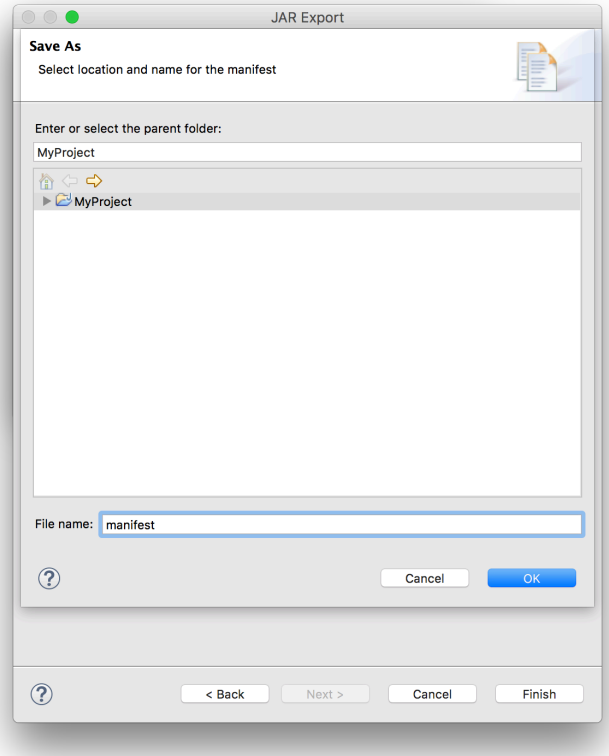
We need to input the name of the class to run when the JAR file is launched. Near the bottom of the window, select the **`Main class`** using the **`'Browse...'`** button. The main class (**`MyClass`**, in this case) should show up in the list if you correctly added the **`main()`** method as described previously.

If you are exporting a project without any other JAR files (e.g. without the ACM libraries), then *just click "Finish"*. You're done! Go ahead and send the JAR to family and friends. Just remember that they need to have Java installed too; see the end of this handout for more information. **If you are exporting a project with other JAR files** (e.g. with the ACM libraries), continue on to the next step.

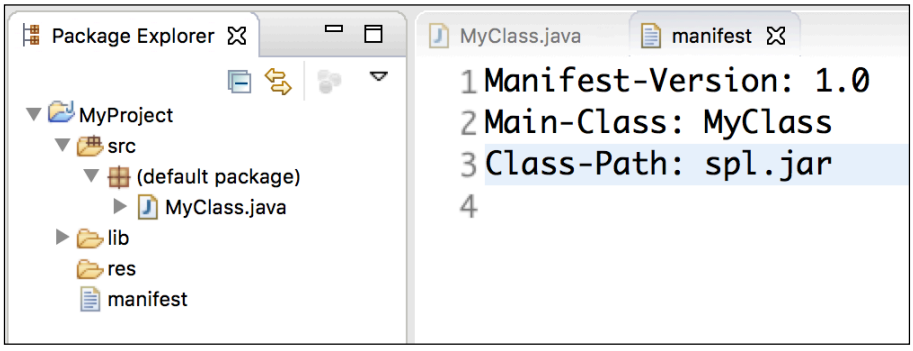
6. Now we need to create a manifest file. To do this, we will go through this exporting process *twice*. The first time is to generate a manifest file, and the second time is to use that file when exporting our program. So, from here, make sure the **'Generate the manifest file'** radio button near the top of the window is selected, and that check box for **'Save the manifest in the workspace'** is checked. Click the **'Browse…'** button associated with the **'Manifest file'** text box to select the destination of the manifest file.



7. In the window that appears, click the **MyProject** folder (or whatever your project is named), and then in the text box type in the name **"manifest"** and click "OK" to finish. The Manifest file path should now appear as something like /**MyProject/manifest**. Click the **'Finish'** button.

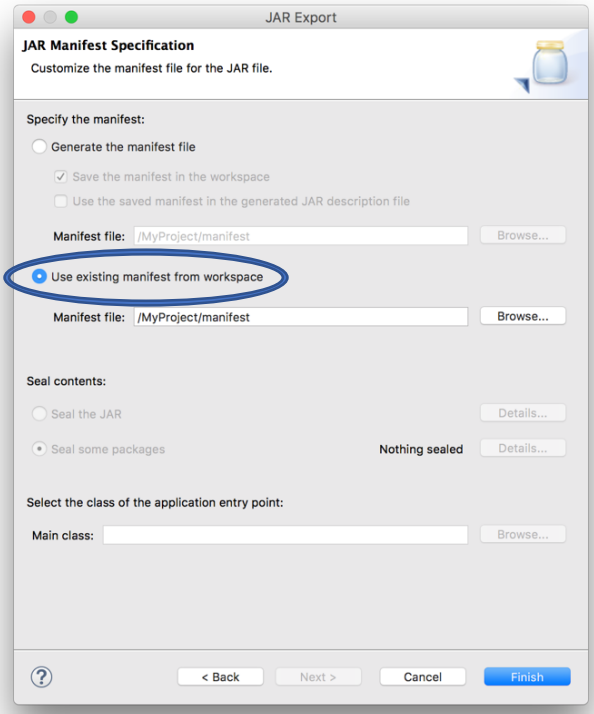


8. Now you should see the manifest file show up in the Eclipse sidebar. Double-click it to open it. You need to edit the manifest file to add the line **`"Class-Path: "`** followed by the names of all the JAR files that this program uses, separated by spaces. In this case, that would include **`spl.jar`**, since that contains the ACM libraries. When you're done the manifest file will look something like this:



Make sure to save the updated manifest file. Now that we have this manifest file, repeat the entire above process of exporting a JAR file (i.e., click on your project name, pick **`Export...`** from the file menu, select the JAR file option for exporting, etc.). However, this time you will do something different when you get to the last window, as explained in the next step.

9. When you get to the last screen, you do not need to generate a manifest file or specify the Main class. Instead, just make sure to click the radio button for **`"Use existing manifest from workspace".`** Now, hit the **`"Finish"`** button. Eclipse will use the manifest file we just created previously to make our JAR file. If it asks you to overwrite the old JAR file, just say "Yes".



10. Now you have your JAR file containing your project code, but unfortunately you can't simply send it to your friends. This JAR doesn't contain the code in the other JAR files it relies on (e.g. **`spl.jar`**), nor does it contain any data files your program might use (text files with data, or even sounds or images). What you'll want to do is create a new folder, place your exported JAR file in it, along with any other JAR files your program uses (like **`spl.jar`** and any other the ones you added to the **`manifest`** file) and data files you use. Make sure to put your data files in a folder called **`res`** if your code expects that. Once you have all of these files in a single folder, you should be able to just double-click your JAR file, and have it run your application. You will need to distribute this *entire* folder to anyone that you want to share your program with. Usually the easiest way is to just zip the folder up into a single file, and then email that. Then, the recipient can unzip the folder and double-click the enclosed JAR file to run your program!

A final note regarding the Java Runtime Environment (JRE)

Anyone who has your JAR files does not need Eclipse to run it, but they will need to have the Java Runtime Environment (JRE) installed on their computer. In fact, you may have installed the JRE on your own computer at the beginning of CS106A so that you could work with Java. Some, but not all, computers come pre-installed with the JRE. We just wanted to point this out in case you pass along the JAR files for your program to someone who may not have the JRE installed and was having problems trying to run your program as a result. Pass along the instructions on the CS 106A website to install the JRE.