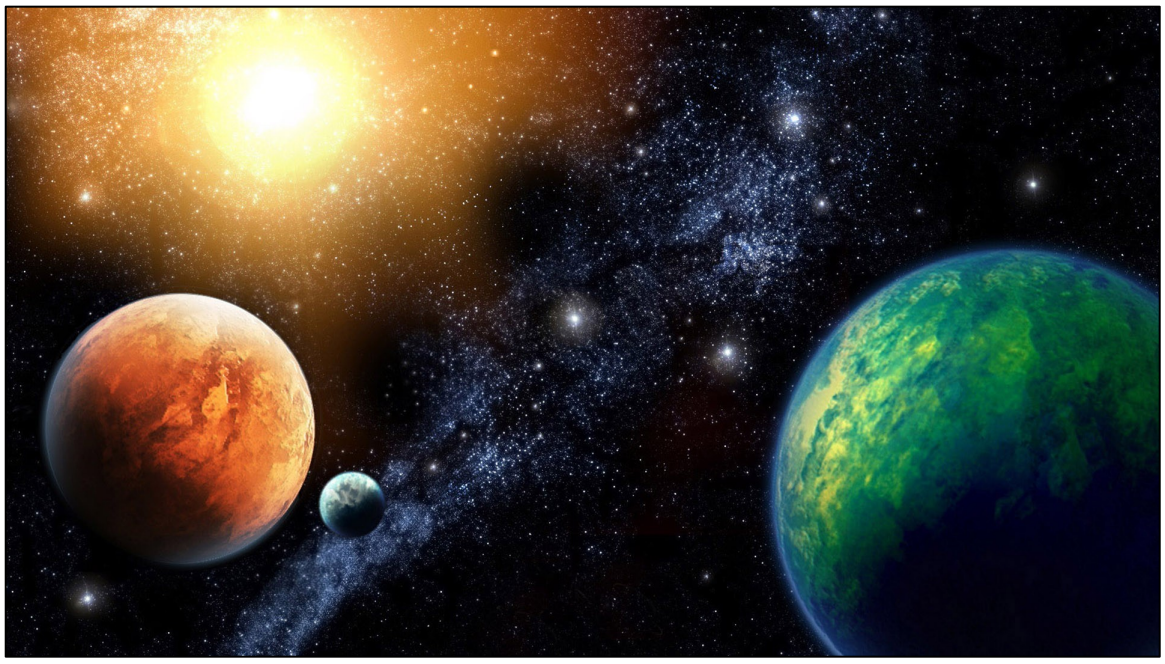
CS 46B
Spring 2017
Homework 1 / Lab 1



In this assignment you will write a very simple program that computes the average mass of the 8 surviving planets in the solar system.

The point of this assignment is just to get you familiar with the process of downloading an assignment, creating an Eclipse workspace, completing the assignment, testing your code, and uploading your solution. It sounds complicated but it isn't, and you'll soon get used to it. If you took CS 46A at SJSU, this semester's procedure is simpler.

Hopefully you will finish everything during lab. If you have trouble, work with your lab partner and upload your solution by the end of the day on Feb 8.

Lab Rules

Work in pairs. One of you will be the "driver", who types into Eclipse and submits a simple lab report. The other is the "scribe", who submits a more detailed report. Alternate jobs, week by week.

You will both submit lab reports.

Driver: At the top of your lab report write

CS 46B Lab Report
Your name
I was the driver, Xxxx Yyyy was the scribe.

Scribe: At the top of your lab report write

CS 46B Lab Report
Your name
I was the scribe, Xxxx Yyyy was the driver.

As you work through the lab assignment, instructions/questions highlighted in yellow should be discussed by both of you and addressed by the driver in the driver's report; instructions/questions highlighted in blue should be discussed by both of you and addressed by the scribe in the scribe's report. Type your answer under a header that gives the step number in the assignment doc.

Step 0: Download the assignment from Canvas

Obviously you've already done that.

In future, lab and homework assignments will be different, but will look similar. You will download a zip archive containing instructions, and possibly starter files and data files. For homework, the zip will also contain an autograder.

Homework autograders will be the same bots that will evaluate your submitted homework assignments. So you can test your work before you submit it. Lab assignments have no autograders. In the real world, you will be responsible for determining when your code is working correctly; you won't get an A for writing correct code … your company fails if you write incorrect code.

Step 1: Download and install Eclipse

If you don't already have Eclipse installed on your laptop, download it before your lab session. Browse to https://eclipse.org/downloads/eclipse-packages/ and download the "Eclipse IDE for Java Developers" for your OS. (Note: NOT "Eclipse IDE for Java EE Developers".) To choose your OS, use the menu labeled "Eclipse Neon (4.6) Release for" at the top of the page. Follow the installation instructions.

Step 2: Create an Eclipse workspace

An Eclipse workspace is just a directory. You have to create it yourself. It's a good idea to put "workspace" or "eclipse_workspace" – or something like that - in the directory name.

Step 3: Start Eclipse

Double-click on the icon to launch Eclipse. If this is a new installation, you will see a Welcome screen (Fig. 1). Otherwise you will see the Workbench screen (Fig. 2). If you see the Welcome screen, click the "Workbench" button (circled in red).

Fig. 1 – Welcome screen.

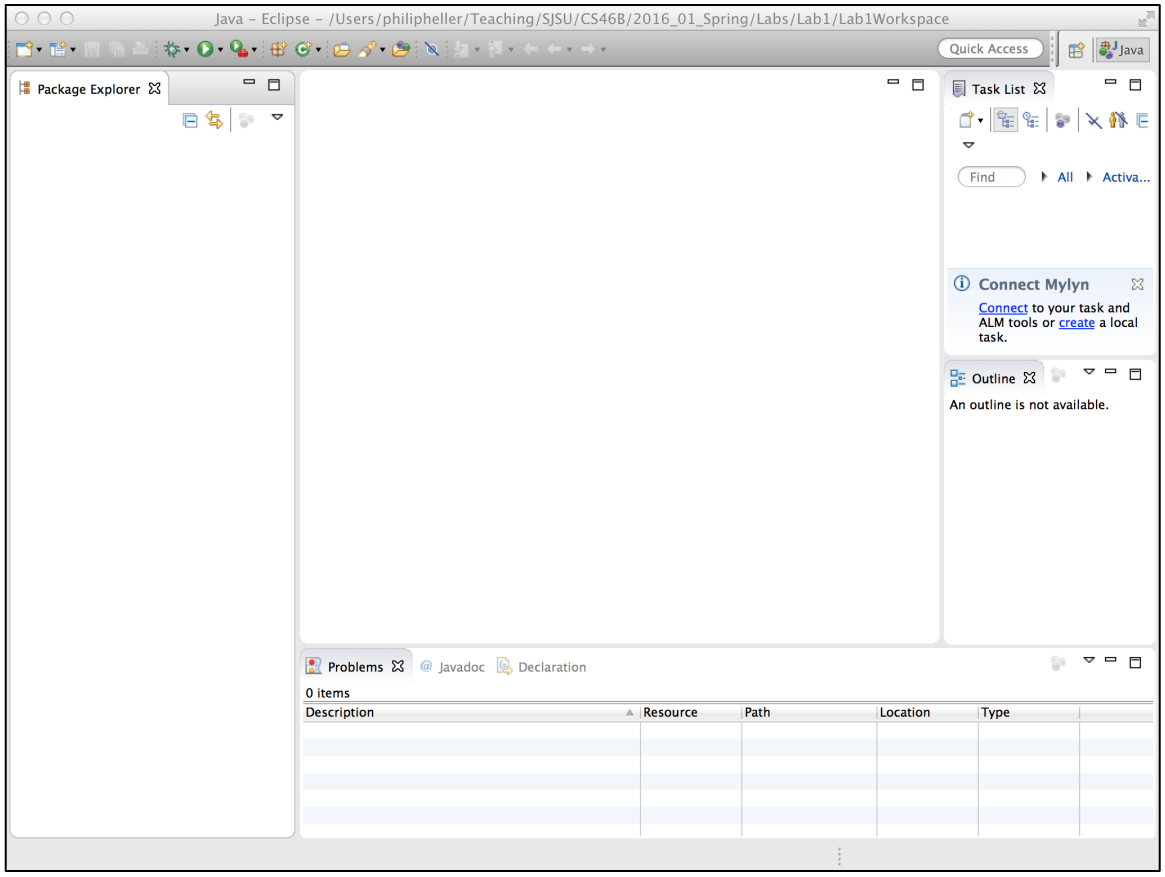


Fig. 2 – Workbench

The workbench contains lots of helper panels that add clutter. For now you can get rid of all the panels except the Package Explorer on the left and the large empty rectangle in the center (this is where your source files will appear).

Step 4: Create a new Java project

On a Mac, use the File -> New -> Java Project menu. On a different machine, your lab instructor can help you figure out how to do that if it isn't obvious. Caution: create a new Java Project, not a new Project. A wizard will appear. Type "hw1_proj" for the project name, and use defaults for all other settings. When you're finished, the Package Explorer will display your project (Fig. 3).

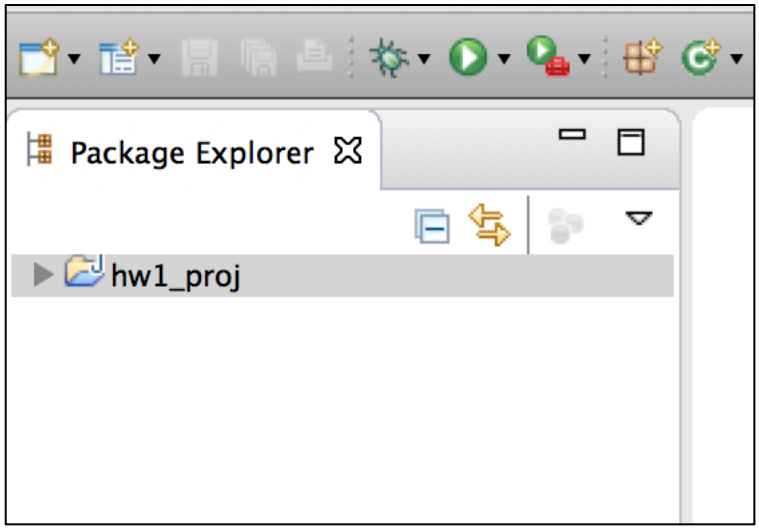


Fig. 3 – After creating the project

Click on the tiny right-arrow next to the project name to expand the project (Fig. 4). There isn't anything in the project yet, so the expansion won't be dramatic.

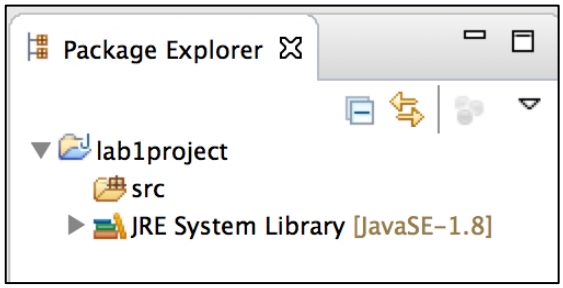


Fig. 4 – Expanded project

Step 5: Create a planets package

Right-click the src icon in the Package Explorer, use the New->Package menu item, and create a package called “planets”. If you’re not on a Mac, figure out how to do this or ask you lab instructor if you get stuck. The package should appear as a member of “src” (Fig. 5).

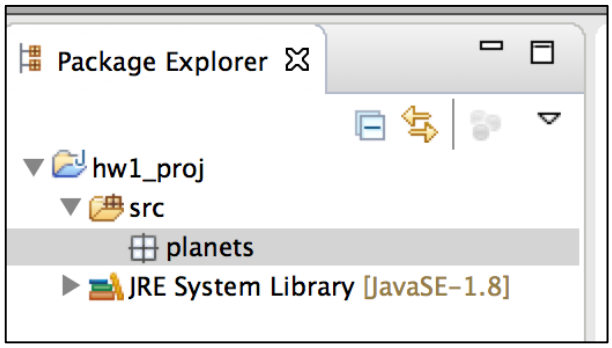


Fig. 5 – Expanded project, after creating package

Step 6: Add class Planet to planets package

The assignment zip archive contains source file Planet.java. Drag its icon into the box icon to the left of “planets” in the Package Explorer. Be sure the tip of your pointer is in the box icon before you release the mouse button. You’ll see the dialog in Fig. 6. Make sure “Copy files” is selected and then click “OK”.

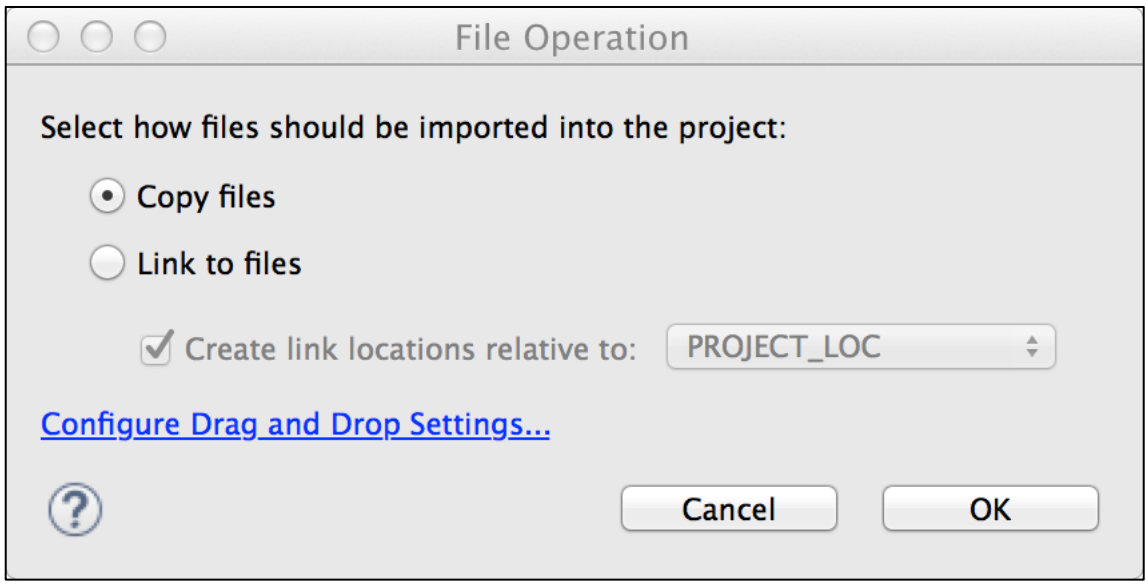


Fig. 6

The Package Explorer should now look like Fig 7. If it doesn't, you probably copied Planets.java to the wrong destination. Delete it and try again.

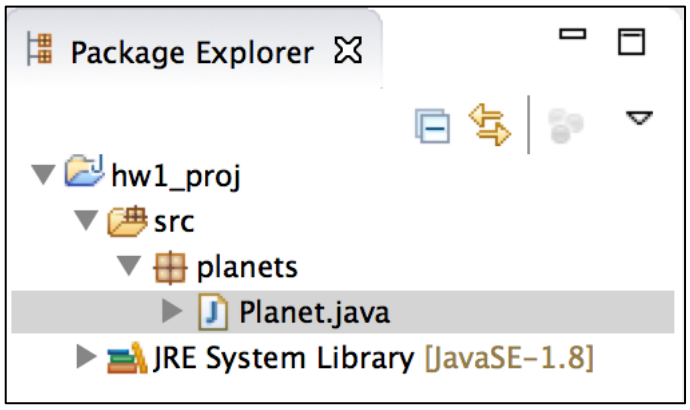


Fig. 7

Double-click on "Planet.java" in the Package Explorer. The source code will appear in the main Eclipse window.

Also drag MassAveragerStarter.java and Homework1Grader.java into the planets package. These sources are incomplete so they don't compiler correctly and you'll see red flags on their icons in the Package Explorer. That's ok – your job is to make them work.

Step 7: Rename MassAveragerStarter.java

In Eclipse, "refactor" means "rename, with benefits". Later you'll see what this means.

Right-click on "MassAveragerStarter.java" in the Package Explorer and click Refactor -> Rename in the popup menu. In the dialog that appears, type "MassAverager" into the "New Name" field and click "Finish". The class name will change in the Package Explorer.

Step 8: Look at Planet.java

Double-click on Planet.java in the Package Explorer. The main panel will display the source code.

Look at the comment at line 5. It cites the source of the data in the array below. In a serious university, everyone must cite the source of text, code or data that someone else wrote or provided. If you don't do that, you are implicitly taking credit for something you didn't do. SJSU expects all students to honestly cite all sources. I also expect this. Consequences for not citing (also called *plagiarism*) range from failure of the assignment through failure of the course, suspension, or expulsion from the university.

Scribe: Do you understand what plagiarism is? Do you promise not to do it?

Driver: Do you understand what plagiarism is? Do you promise not to do it?

Both: Write your answer in your lab report, under heading "Step 8". In all labs, whenever you are instructed to write something in your report, use "Step #" headers.

Driver: type some nonsense words into the source code. Notice how Eclipse immediately flags the bad code. Eclipse automatically compiles your code whenever you change it; you don't have to push a button in order to compile. Scribe: where does Eclipse mark bad code? What happens when the mouse cursor is moved onto the "bad code" mark?

Participation points opportunity … You read the syllabus, right? If not, read it today, after lab. It records the rules for CS 46B, and you are expected to understand it. It says that part of your grade is for participation on Piazza. You should post to Piazza a few times per week or more. Here's your chance. The instance variable "massKg" is a good name. Now or any time this week, post a message to Piazza (in folder "mass") explaining why it's a good name.

Step 9: Fix MassAverager.java

Implement the getMeanPlanetaryMass() method in MassAverager. Just fetch the array of planet objects, compute the sum of the masses, and divide by the size of the array.

Scribe: you know the array size is 8 because you just looked at the Planet.java source code. Why is it better to divide the sum-of-masses by applying ".length" to the array, rather than literally dividing by 8?

To execute your code, left-click on "MassAverager" in the Package Explorer. Make sure it is highlighted. Then click the tiny "Run" button (white arrow in a green circle) in the top-row toolbar. A console panel will appear, containing your output.

Step 10: Test your solution

To test your code, left-click on "Homework1Grader" in the Package Explorer. Make sure it is highlighted. Then click the tiny "Run" button (white arrow in a green circle) in the top-row toolbar. The grader bot is almost exactly the same app that the human graders use to score your work; the only difference is that the human graders also check your code for style and comments.

Your future homework assignments will contain specific instructions about how to implement your code (for example, you might be told what class or method names to use). It's possible to write code that computes the right answer but doesn't follow all the assignment instructions. The grader bot won't be able to evaluate such code, and the code will get a bad grade.

To better understand how this works, rename the getMeanPlanetaryMass() method in MassAverager. Just edit the source, and type "x" somewhere in the name. Hmm, now the code doesn't compile, because lines that call getMeanPlanetaryMass() are now calling a nonexistent method. Change the method name back.

You need a better way to rename a method. *Refactoring* is Eclipse's better way. Double-click on "getMeanPlanetaryMass" in the method declaration to highlight it, move the cursor into the highlight, and right-click to bring up a large menu. Select Refactor →Rename. You will be prompted to type a new name for the method; type any name you like, then type Return or Enter. Notice how the name changes both in the method declaration and in the last line of main().

Run MassAverager as you did in Step 9. Obviously it prints the same answer as before. But … run Homework1Grader again. Scribe: what happened to your score? It seems unfair to deduct so many points just because you gave a method a different name, but the bot needs the name to be exactly what it expects. You should never lose a ton of points for this kind of reason, because you will always be able to run the bot before you submit your work.

Change the method name back to "getMeanPlanetaryMass", verify that MassAverager runs correctly, and rerun the grader bot.

Step 11: Export your solution

Now you'll create a jar archive containing your source code. Right-click on "src" in the Package Explorer. Make sure it is highlighted. IMPORTANT: If you click on the wrong line, the following steps will work but your submission will have the wrong

contents. You are responsible for doing this step correctly; if you submit the wrong contents, you won't get a second chance to do it right. So always do this step carefully, and then check your contents as described below.

When you right-click on “src”, you’ll get a popup menu. Select “Export…” to bring up an export wizard (Fig. 8). If you don’t see “JAR file”, “Javadoc”, and “Runnable JAR file” under “Java”, click on the little triangle to the left of “Java”.

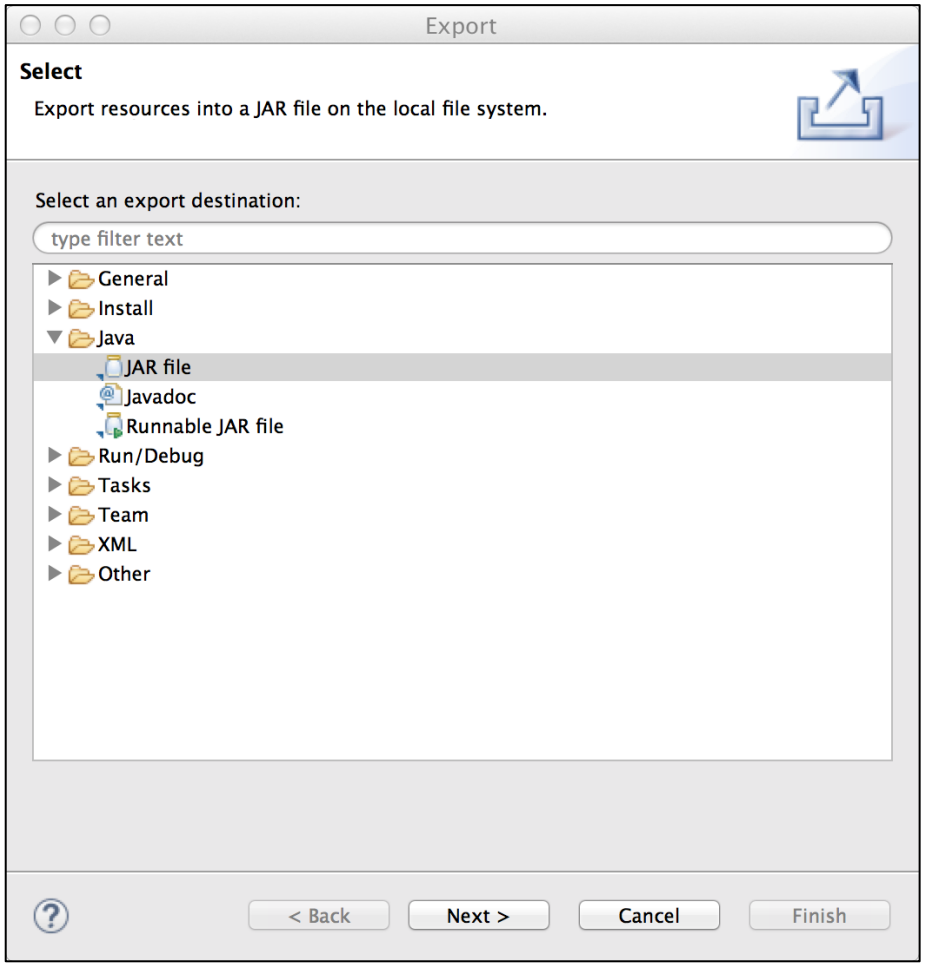


Fig. 8 – Export Wizard

Click on “JAR file”. Important: don’t click “Runnable JAR file”, which would export the wrong contents, and you are responsible for blah blah blah and you’ll get a bad grade and it’s not negotiable so don’t do it.

Click “Next >” to bring up another wizard (Fig. 9). For the export destination, name your file Homework1.jar and specify some convenient directory (home or desktop are good). Configure the rest of the wizard exactly as shown in Fig. 9:

- Don’t select .classpath or .project

- Check “Export Java source files and resources” and “Compress the contents of the JAR file”; don’t check anything else.

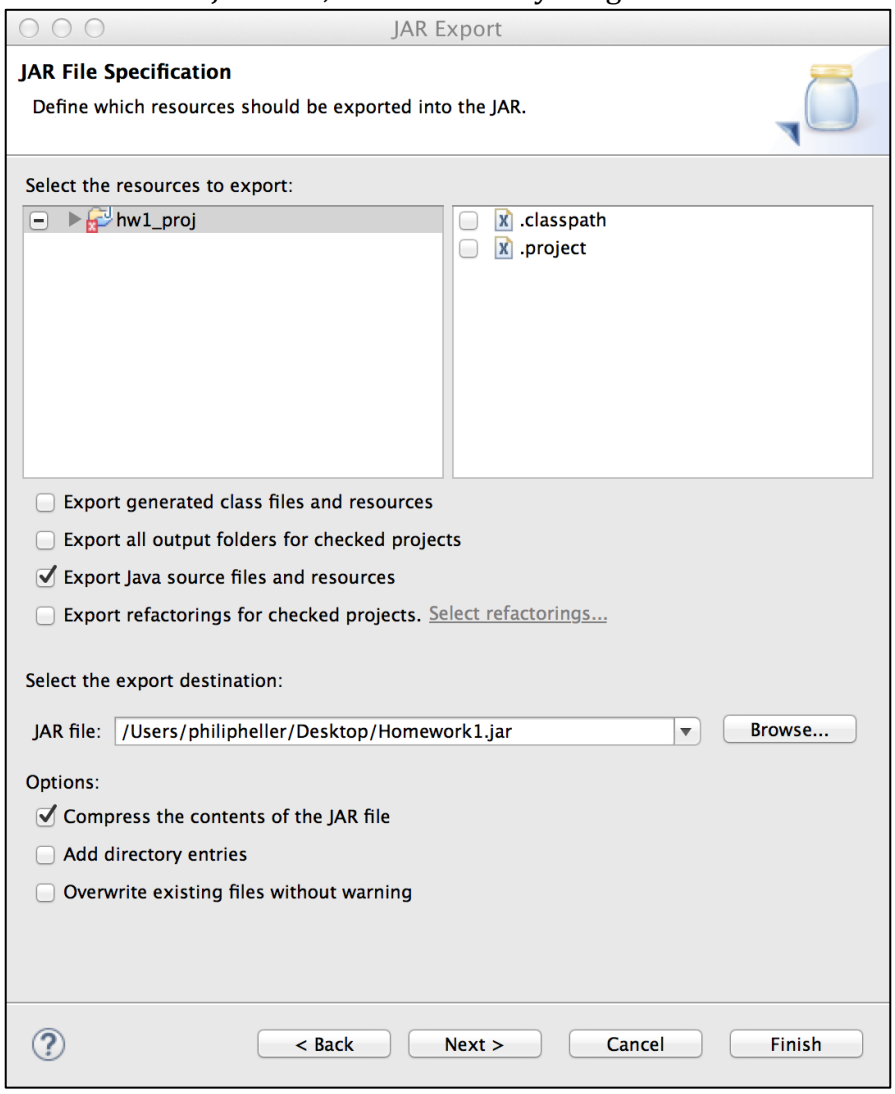


Fig. 9 – 2nd Export Wizard

Step 12: Check Homework1.jar

Open a terminal/console/command window, and cd into the directory where you stored your Homework1.jar file. Type “jar tvf Homework1.jar”. You should see something like the following:

```
7738 Wed Jul 06 09:03:34 PDT 2016 planets/Homework1Grader.java
 430 Wed Jul 06 09:02:58 PDT 2016 planets/MassAverager.java
 942 Wed Jul 06 08:25:46 PDT 2016 planets/Planet.java
```

If you see different output (other than datestamp and the 1st number on each line, which is the file size) , delete the jar file and export again. Scribe: does the output correspond to what you see in Eclipse's Package Explorer?

For future homework assignments, this step will be your final check to make sure your jar contents are correct. You are responsible for the contents of any jar you submit. Any homework submission that doesn't contain all the required Java sources will receive zero points.

Step 13: Upload Homework1.jar

Driver: open a browser, go to your Canvas page, and upload Homework1.jar.

Scribe: Using the Driver's computer, upload Homework1.jar to your Canvas page.

Ok, that's how you do homework in CS 46B. From now on, your lab assignments will be different from your homework assignments, and you won't upload your lab assignments.

Step 14: Check out with your lab instructor

Since this is a lab as well as a homework assignment, you have to follow lab procedure as described by your lab instructor at the start of this session.

If you finished early, ask your instructor if you can leave.

If you ran out of time: there is no penalty for not finishing a lab assignment, but this is also a homework assignment, so finish it later and submit it before the deadline.