Submission Instructions

Computer Science 2334

(Revised Spring 2009)

To submit your assignment, you should:

1. Generate the appropriate Javadoc documentation. This will be detailed in Section A.
2. Archive the project as instructed in Section B.
3. Upload the project archive to the appropriate drop box on the Desire2Learn website.

Please follow these instructions carefully. These instructions were developed with both faculty and students in mind. If you have any questions after you read this document in its entirety, please contact your Instructor or one of your TAs immediately. Also, unless otherwise directed, students should submit everything (labs, designs, and final projects) as groups, as detailed in the Documentation Requirements file.

Additional Instructions:

If you are submitting the final project, make sure to include "Milestones.txt" inside the Eclipse project–don't submit it separately. The Milestones file will be included only on the final project submission; not labs or project designs. The Milestones.txt file will list the numbered objectives of the project, and list which objectives you believe that you have met. If you know of problems in meeting the objective, explain the problem, so that you may be considered for partial credit. A sample is given below.

> Milestone #1:
> I met this milestone. The test data which demonstrates this is in milestone1.txt

> Milestone #2:
> I did not meet this milestone.

> Milestone #3:
> I sort of met this milestone. My test data (included in obj3.txt) runs OK until the remove command is done. At this point the program gives a NullPointerException.

By directing the grader's attention to a part of the project that you struggled with, and detailing what you attempted to do (along with what you accomplished) will allow the grader to be more fair in assigning a grade. If you fail to do so, then the grader will most likely assume that you had no idea what was going on, and therefore will not receive as much partial credit.

Section A—Creating the Javadoc

Before proceeding, read about how to generate the correct Javadoc documentation in *Documentation Requirements*. This file can be found on the class website, just as this one is. Java generates nicely formatted, html style documentation for its classes, just like what you see when you browse the Java

API online at ***http://java.sun.com/javase/6/docs/api/***. The trick is that you need to format your comments correctly in order to do so. The Documentation Requirements give those instructions in detail. This document will only show how to generate the documentation after you have properly written your source code.
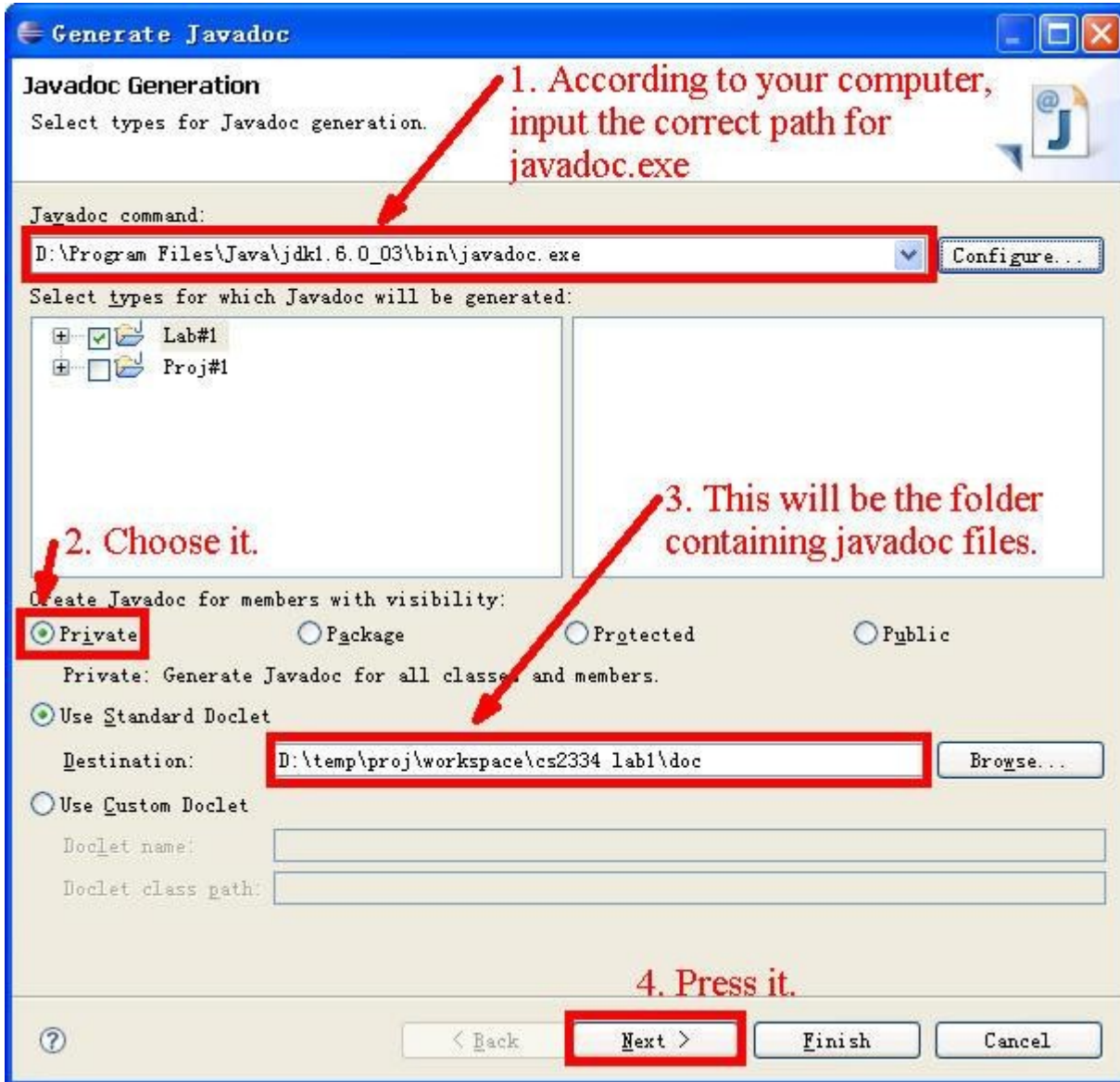
Step 1—Select the project for which to generate the Javadoc
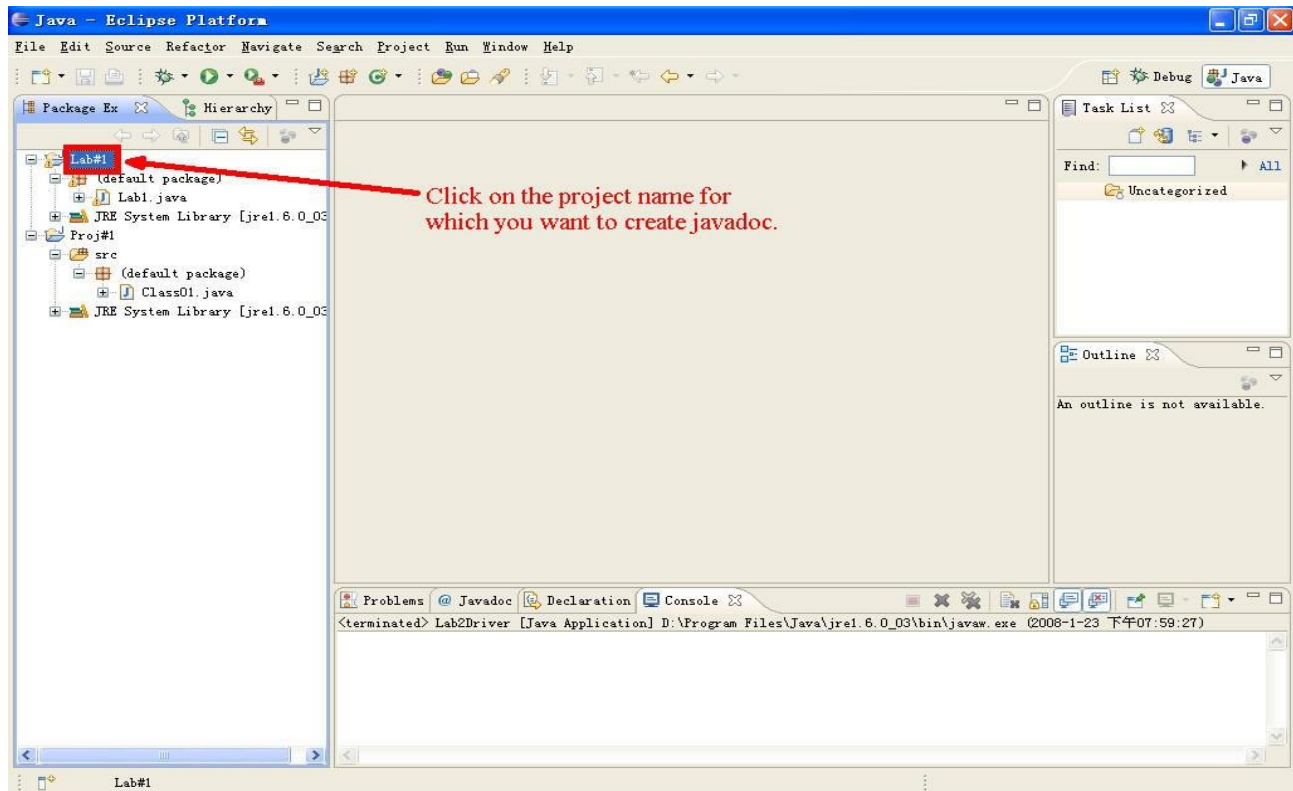
Step 2—Run the command to generate the Javadoc
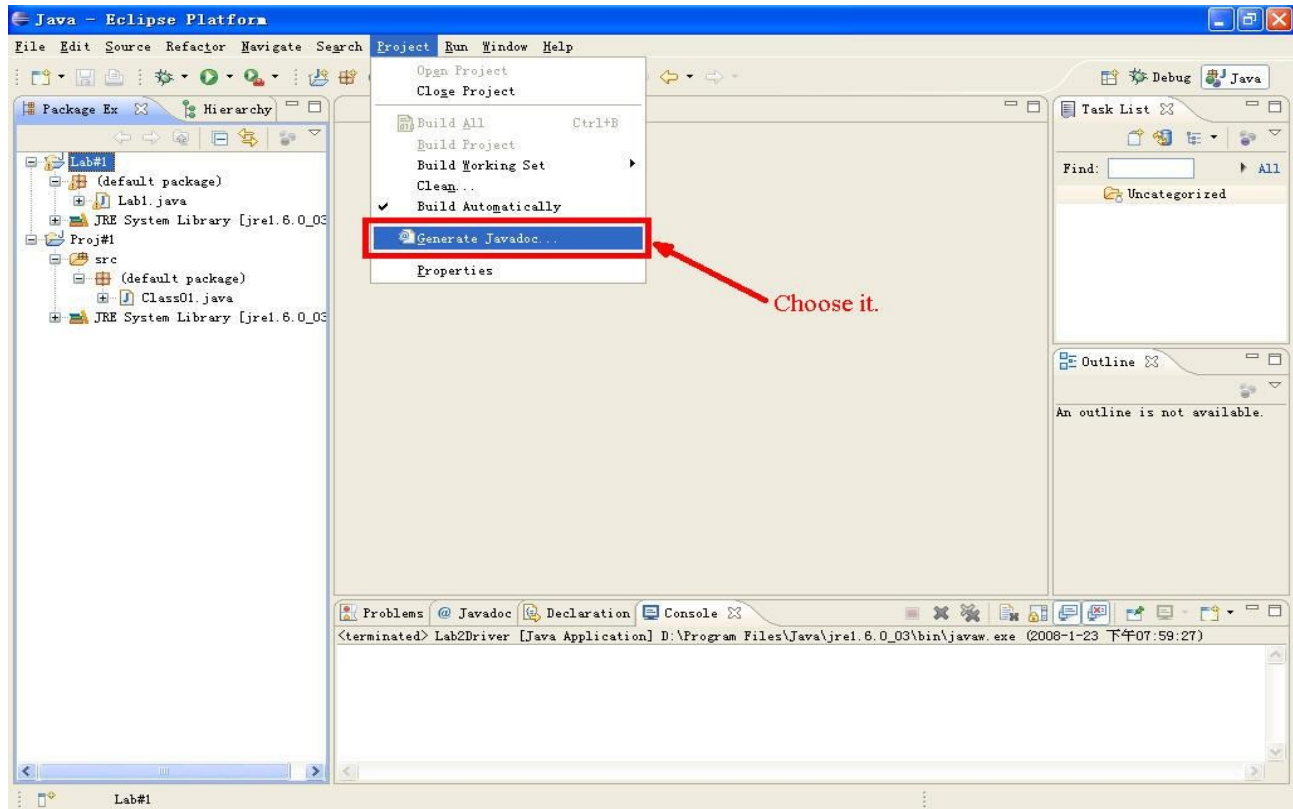
Step 3—Set the correct options

The above window does a few things. First, make sure that the javadoc command that Eclipse is expecting to use is the correct one. The command should be in the same directory as the previously used commands java and javac where, with respect to the jdk that you already installed.

Ensure that you choose to create Javadoc for members with visibility of private. This is not the default, and if you follow good data encapsulation practices (and make most of your class variables private), they will not show up unless you select the "Private" option, as done above. Don't worry, all of your members with public, protected, and package visibility will also be included.

Note the destination location of your Javadoc (#3), as it should default to a "doc" directory inside your project; then select next.

Step 4—More configuration; keep the defaults, except for @author
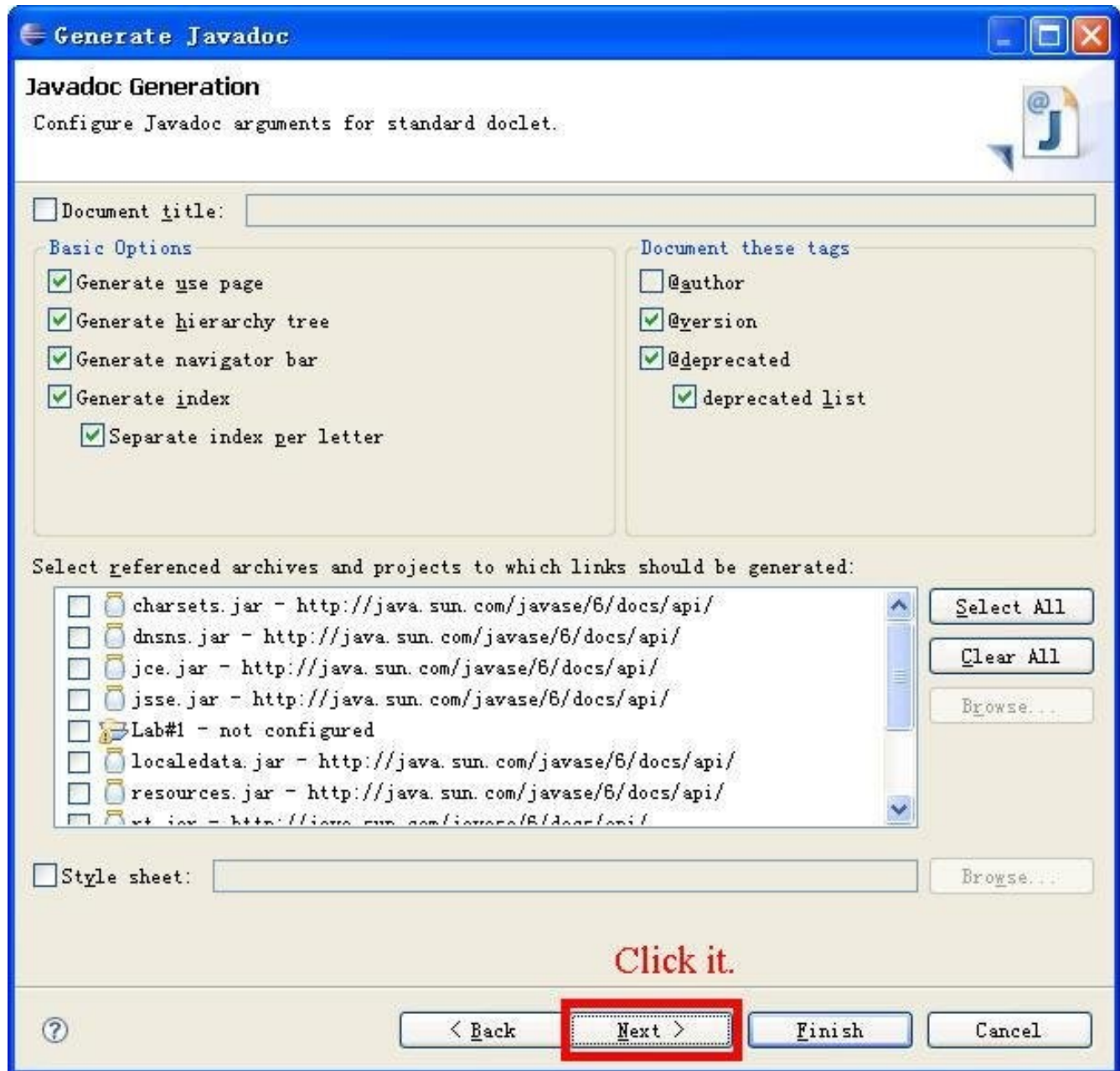
The above window keeps the defaults (or should, at least). Double-check that your selections are the same as above, though, just to be sure.

There is one thing you need to change, however. Since we have to submit a copy of various assignments for accreditation purposes, and we cannot have any personally identifying information on the submitted assignments, we do not want you to include the @author tags. By default, it will be selected, but if you included the tags in your source code (which we will be telling you NOT to do), you need to de-select the option here so that it does not show up. Note that we do not want you to include it on your source code either, so changing this default is really just keeping in line with our overall policy. You will include a cover sheet (to be described later) that details exactly who did what.

Step 5—Format the output and create the Javadoc
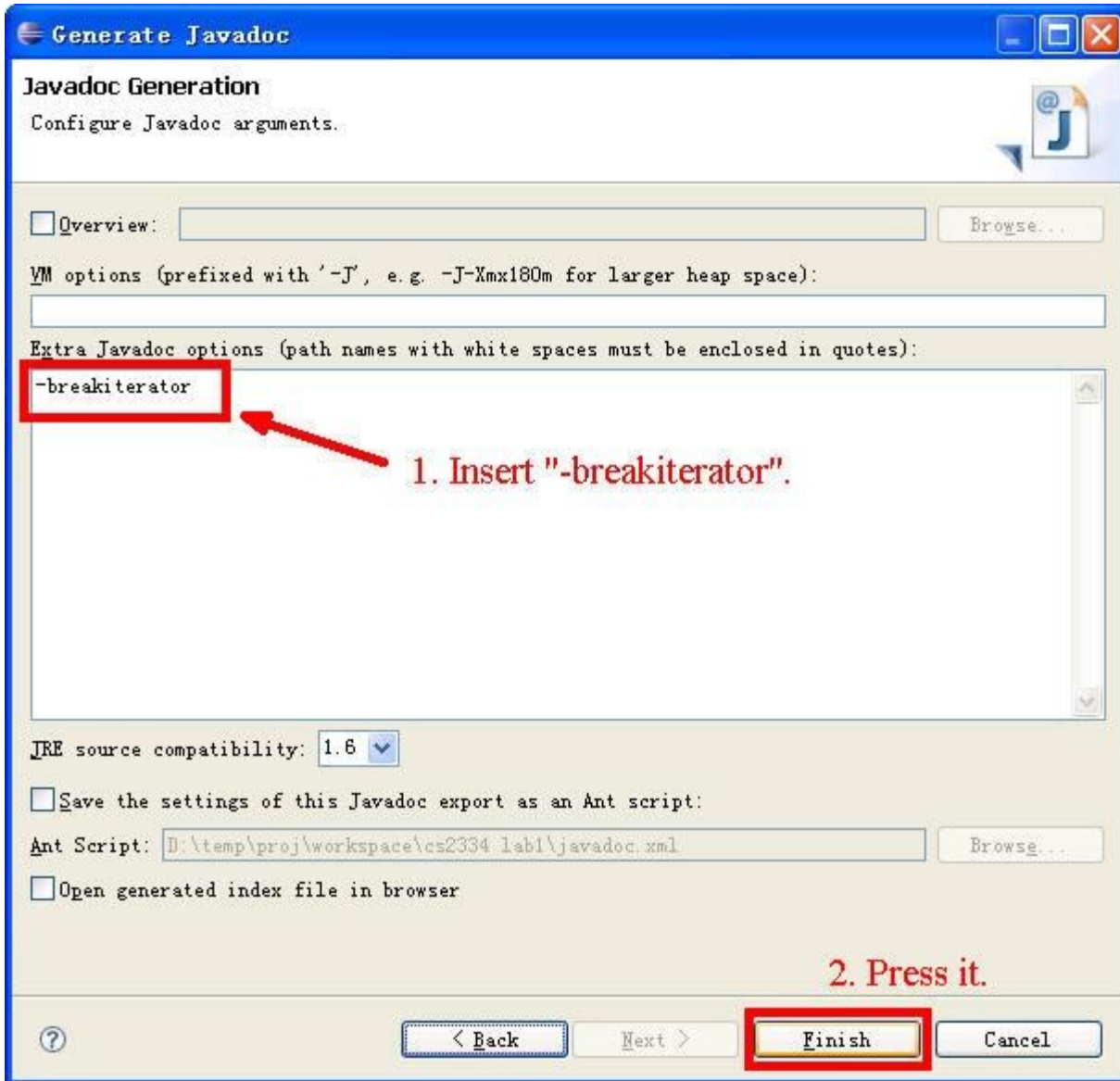
Be sure to insert "-breakiterator" (w/o quotes) as shown above. This will format the Javadoc appropriately. The rest of the options should be the default. Press the Finish button to create the Javadoc.

After running this wizard, you should have the Javadoc show up under a directory named "doc" in your project. Look at your "package explorer" (the left-hand window pane that allows you to browse the contents of any project), and it should be there. You can even right-click on the doc directory, and choose "properties", and discover the full path to your Javadoc, as it resides on the hard drive.

The picture on the next page shows this. This ends our instructions on how to create Javadoc. Please ask a TA if you have any more questions, and do it soon, as we will be creating Javadoc many times this semester, and it is part of your grade. The page after the next picture will begin the steps to properly export an Eclipse project.

This is the javadoc just created.

Right click on it and choose Properties; you will see where is it in your hard drive.
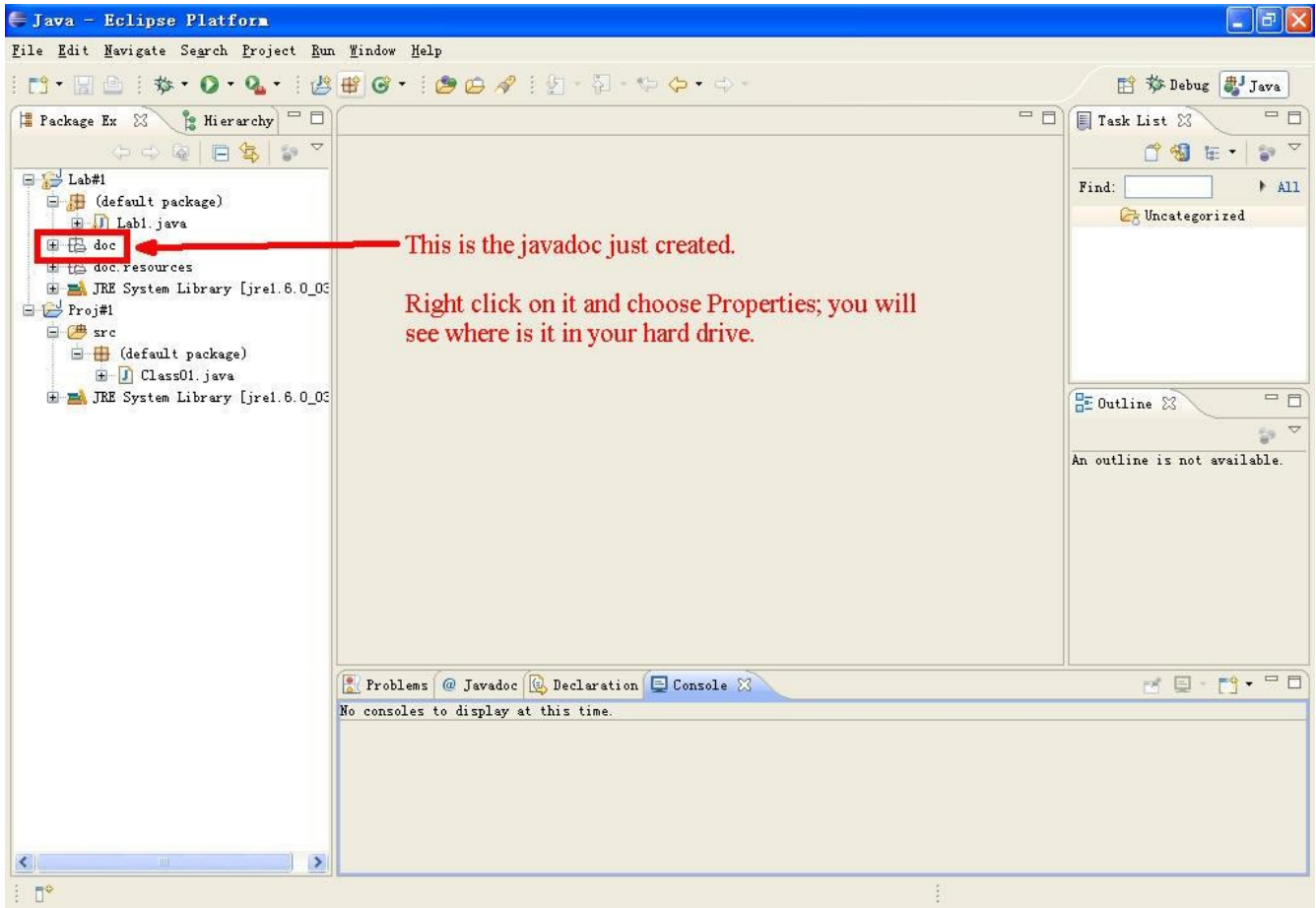
Section B—Exporting an Eclipse Project

The following instructions will walk you through the process of exporting an Eclipse project. We will be submitting all programming assignments via Eclipse archives, so please pay attention to the steps. In addition, the projects will have extra documentation that needs to be submitted, and if it is missing, you will lose points. Please follow the instructions exactly. Before we show how to export an Eclipse project, here is what needs to be submitted for the three types of programming assignments that students will have this semester:

1. ***Lab assignments***—Lab assignments will consist of short and very specific exercises that supplement course material. Often, source code is provided to the student, with the expectation that the student will write code that will enable the existing program to function properly. Detailed instructions are given with each lab. When the lab assignment is finished, the group needs to hand in the "hard copy" (printed handout), and submit the "soft copy" electronically to the appropriate dropbox on the D2L website (***http://learn.ou.edu***). For labs, the soft copy should only contain the project archive (discussed below), and nothing else.

2. ***Project Designs***—Once a project has been assigned, each group will have a specific amount of time to submit both a hard copy and a soft copy of their project design. This design does not need to be final; it may fluctuate and change significantly before the project is finished. What this design should reflect, however, is a carefully thought out plan as to how the project specifications will be reached. This means that a good amount of time should be put in before any code is written. In fact, no code is allowed to be written inside the project design methods. This is called "stubbing" your code. Variables should be declared, and methods should have their signatures (names, and parameters/return types, if not void). A well-though out design should be able to reflect a clear plan as to how the project will be accomplished. The hard copy should contain a printout the cover sheet, stubs, and UML. The soft copy should include the cover sheet, stubs, UML and Javadoc.

3. ***Final Project***—When the final, working project is submitted, the hard copy should contain a printout of the cover sheet, UML, and source code. The soft copy should have all of these, and the Javadoc.

The proper format and content of these items will be elaborated on in the Documentation Requirements file (and in class), but for now, you should concern yourself primarily with what to submit with each assignment, both hard copy and soft. See the next page for the first picture.

## *Step 1—Select and name the project*
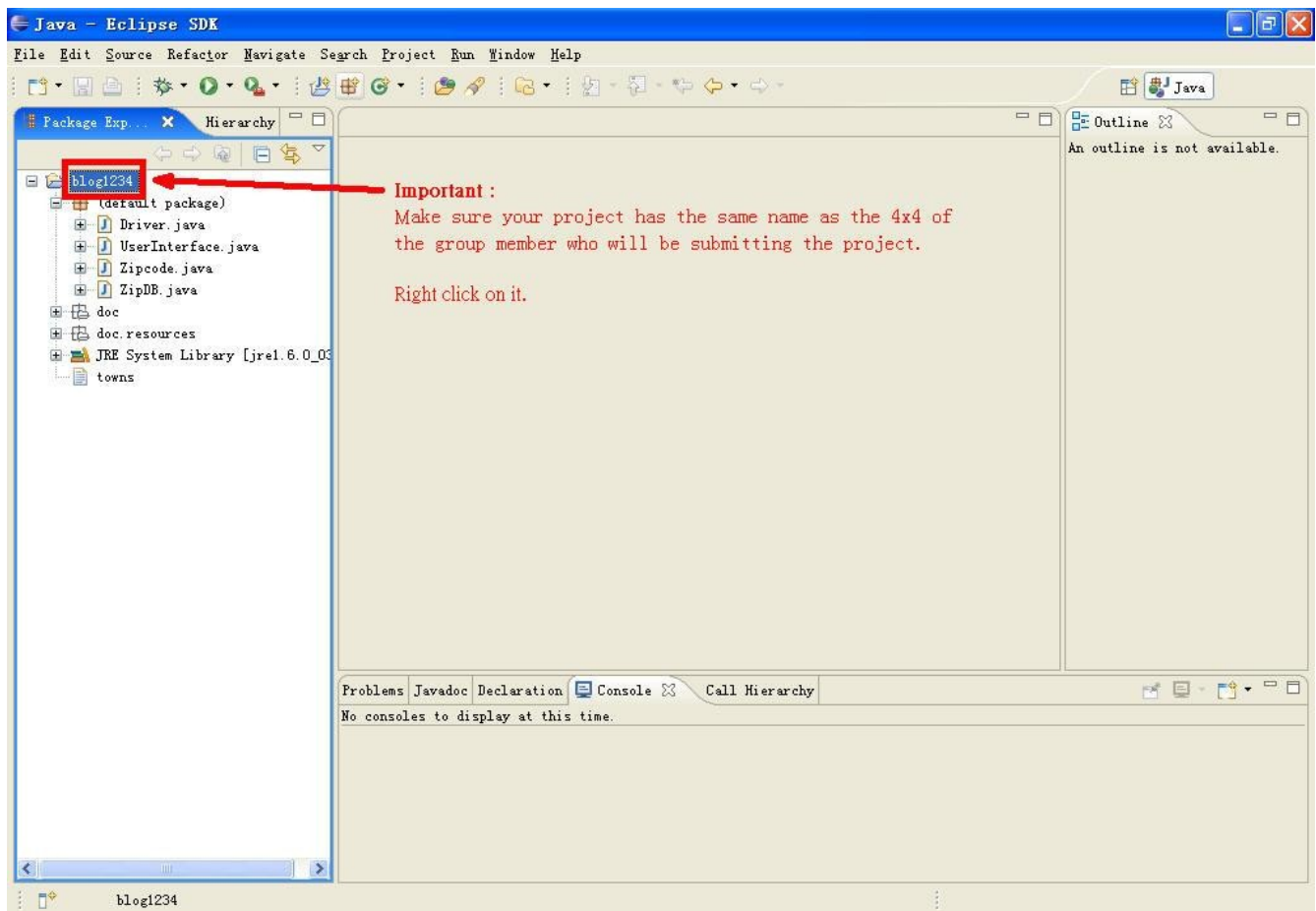
Make sure the project has the correct name, and then select it by using a right click.

Step 2—Choose the export option
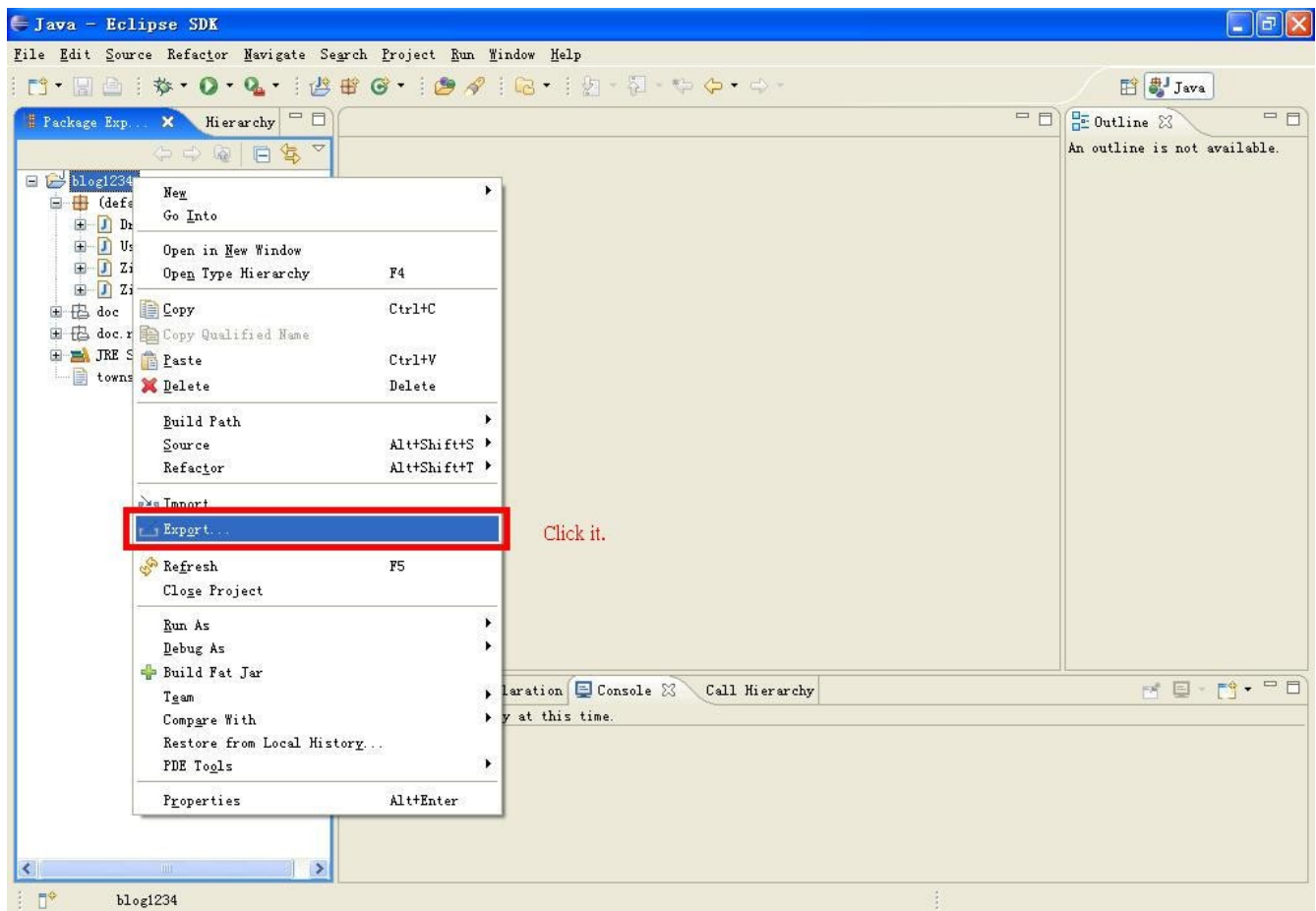
Please use the wizard; do not try to archive the file via a third party application; the grader might not have the same program as you, or the application may not include the hidden configuration files that Eclipse uses to define the project.

Step 3—Choose the correct format
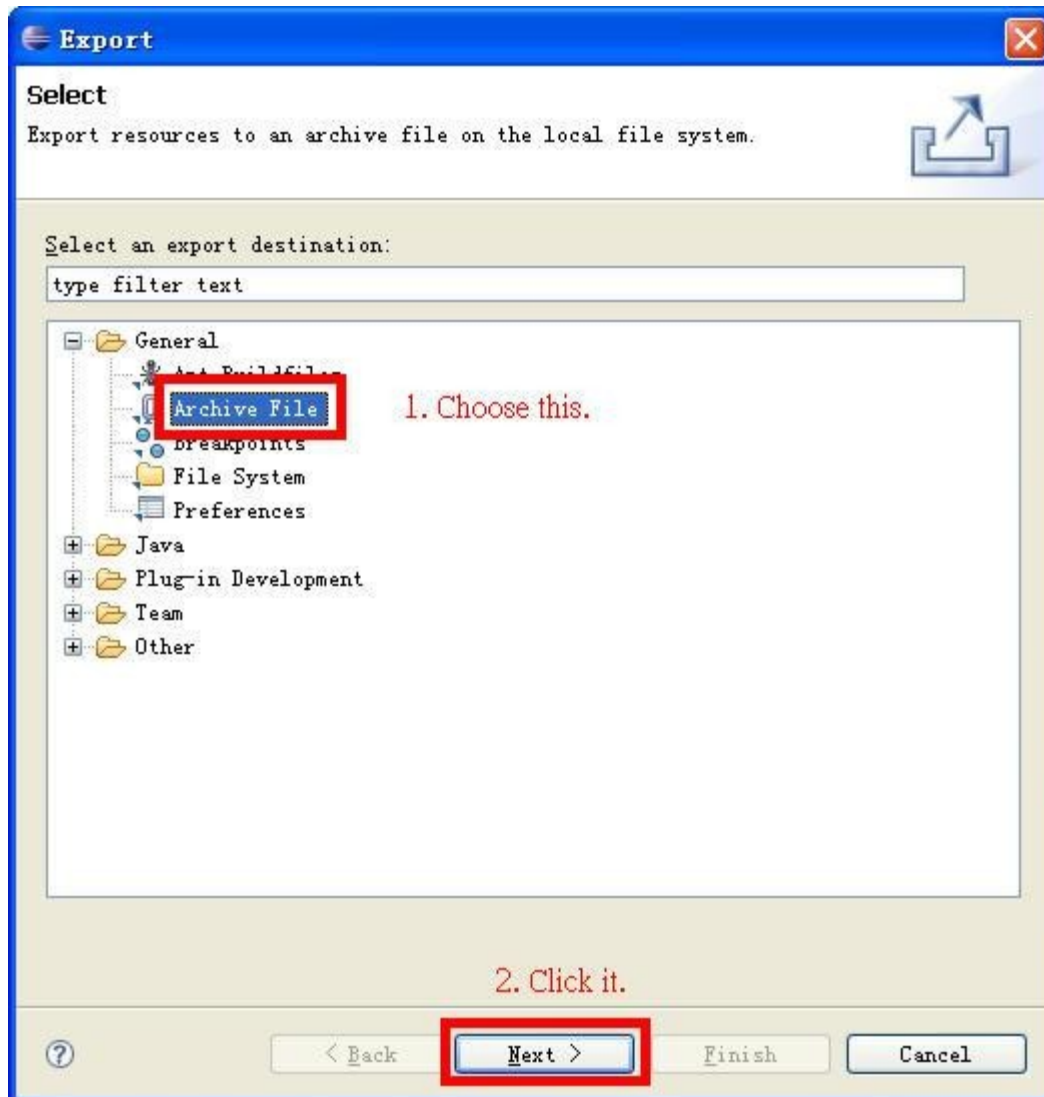
Make sure to choose the archive file; we will be exporting the project to a zip file. Also note that our format falls under the "General" section; not the "Java" section.

Step 4—Choose the destination and run the export
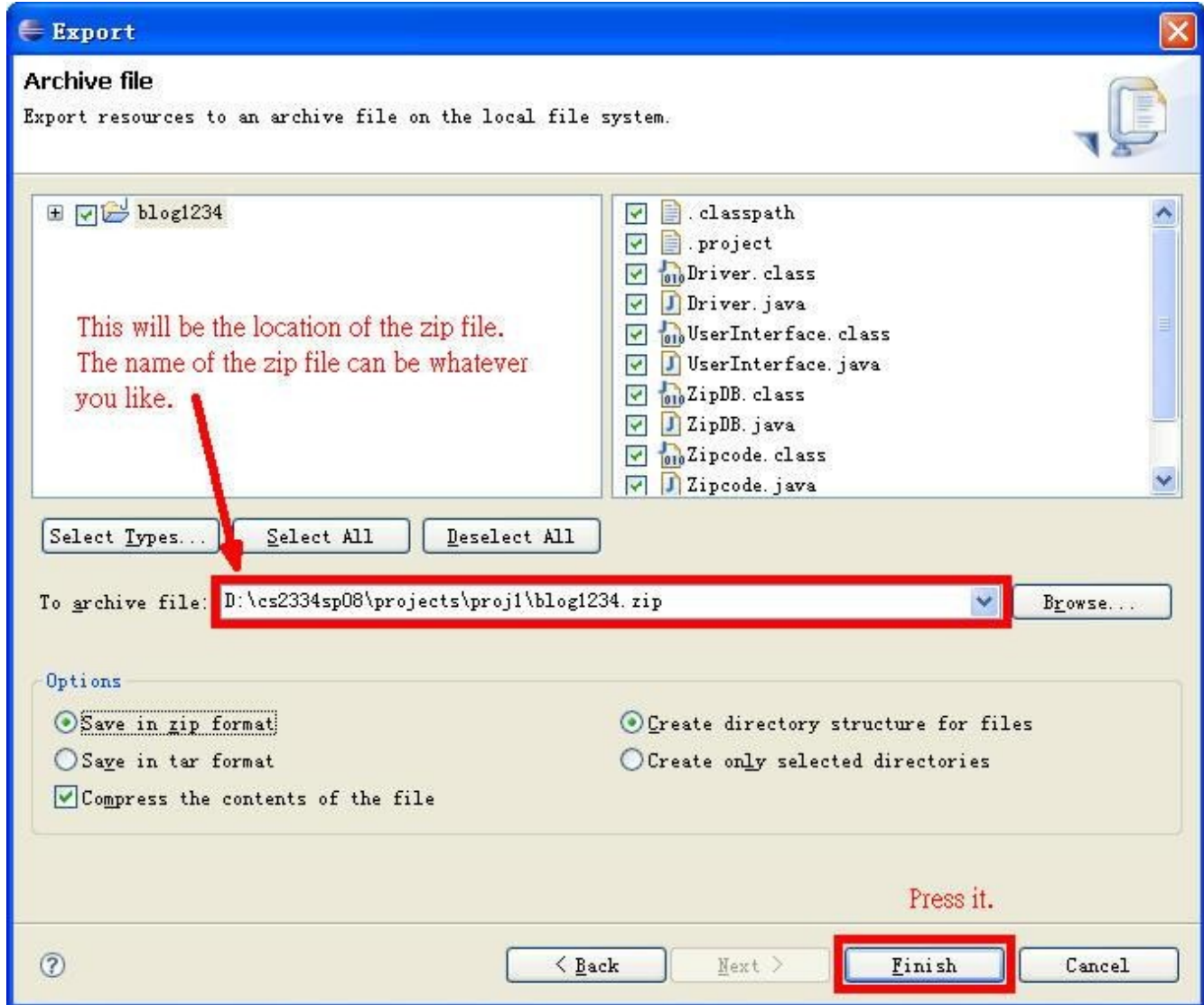
Note that the project we have selected for export is selected for us in the upper-left pane. Also note that all of the files in the project's directory should be selected. This is the point where you should ensure that non-source code files (such as the Javadoc and your UML, and possibly the Milestones file) are also selected for export. Then, choose your destination (it is easiest to choose an easy place to navigate to, such as your Desktop or Documents directory), and remember it, since this is where you will find your zip file to upload to D2L. By default, you should have the zip format selected, and the other options you see above. If not, then choose them. As shown, you can name the zip whatever you like, since once decompressed, it will have a folder that is named the name of your project. Usually, students name it the same as their project. Finally, press the Finish button, and you are done!