




Eclipse Tutorial

Thanks to Sandeep Pasuparth

- 
- In this tutorial we will be walking through a small demo application using the Eclipse Development Environment. Previous knowledge of any kind of tool is not necessary, but can be helpful when understanding why we are following certain steps or how we are making this application work.
 - **www.eclipse.org**

What's Eclipse?

- It is a free software / open source platform-independent software framework for delivering what the project calls "rich-client applications". Eclipse is also a community of users, constantly extending the covered application areas.
- Eclipse was originally developed by IBM as the successor of its VisualAge family of tools.
- Eclipse is now managed by the Eclipse Foundation, an independent not-for-profit consortium of software industry vendors.

Essentials of Eclipse

Before going into Eclipse, some of the basic stuff you need to know are:

- In Eclipse you need to start of with creating a project.
- Then choosing a particular project, we create our java file in it. No need to worry, it is simple and you will learn it by the end of this presentation.
- In Eclipse you need not remember the exact command syntax, it helps you writing the commands.
- Just by saving a file, Eclipse compiles the program by default. It makes work easy for programmers.
- Just go step by step.

Let's start with basic stuff

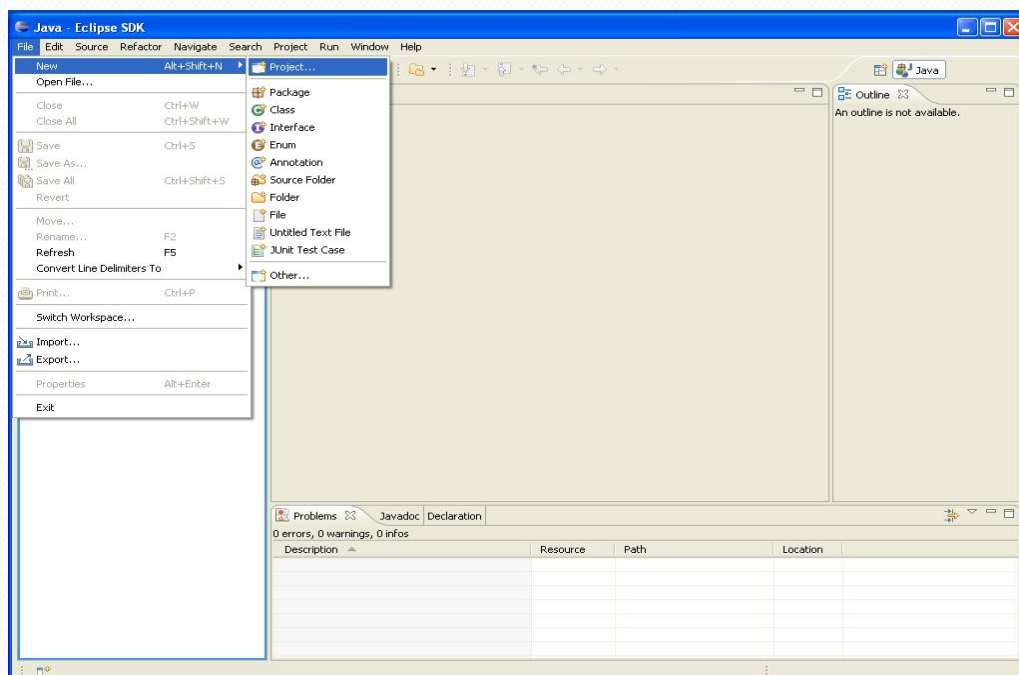
Step1: Open the eclipse from start on your system, choose your workspace



3/3/10

5

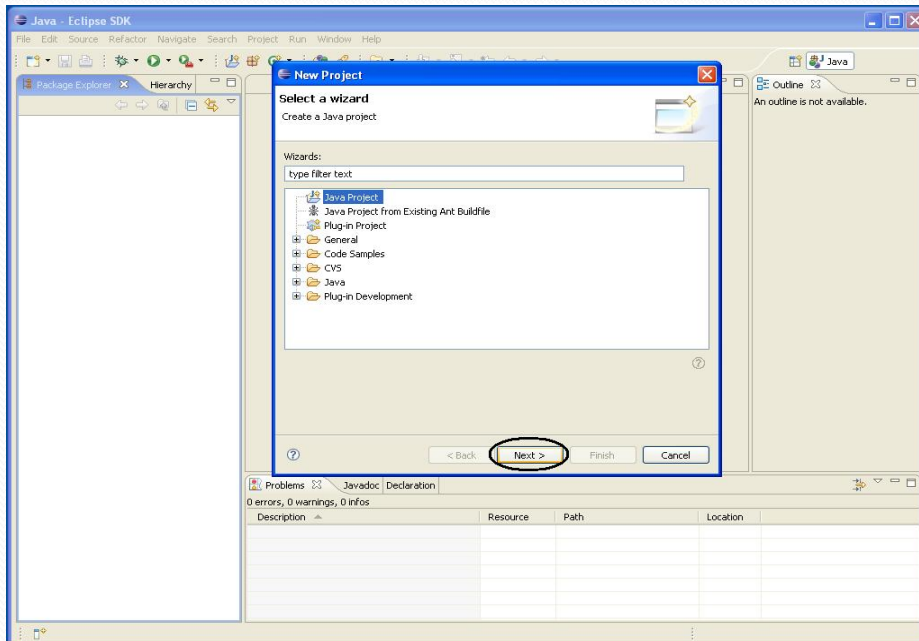
Step2: In eclipse when ever you want to create a class, you need to select the new project by default.



3/3/10

6

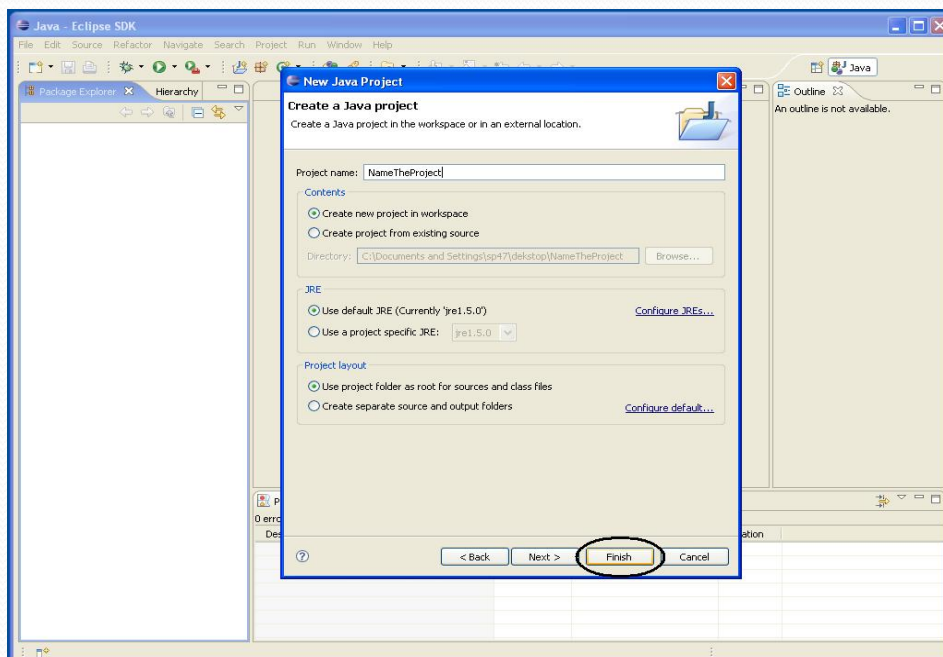
Step3: Select Java project and click next.



3/3/10

7

Step4: Name the project and click finish.

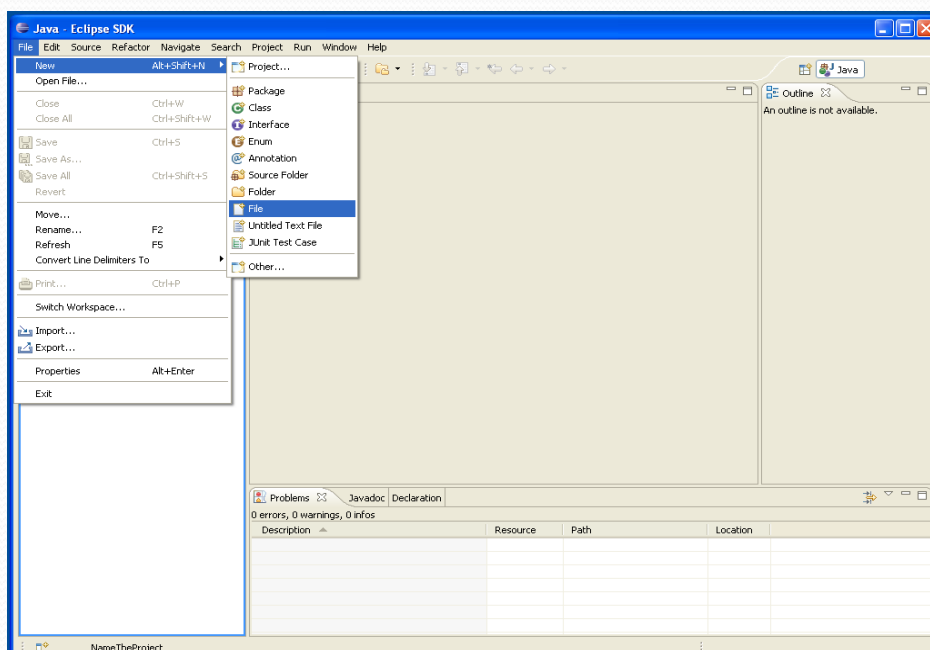


3/3/10

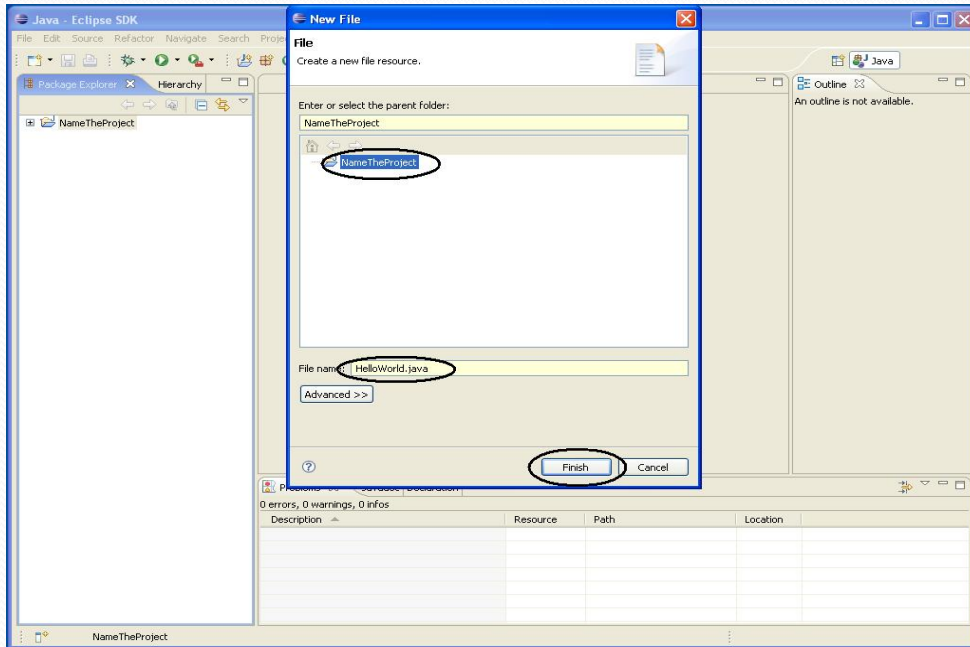
8

- **Now, make a choice...**
 - Create a file directly
 - Import a file
 - Creating/Exporting a .jar file
 - F A Q

Step5: Now we create the java file by selecting the “File” option then “New” and select new file.



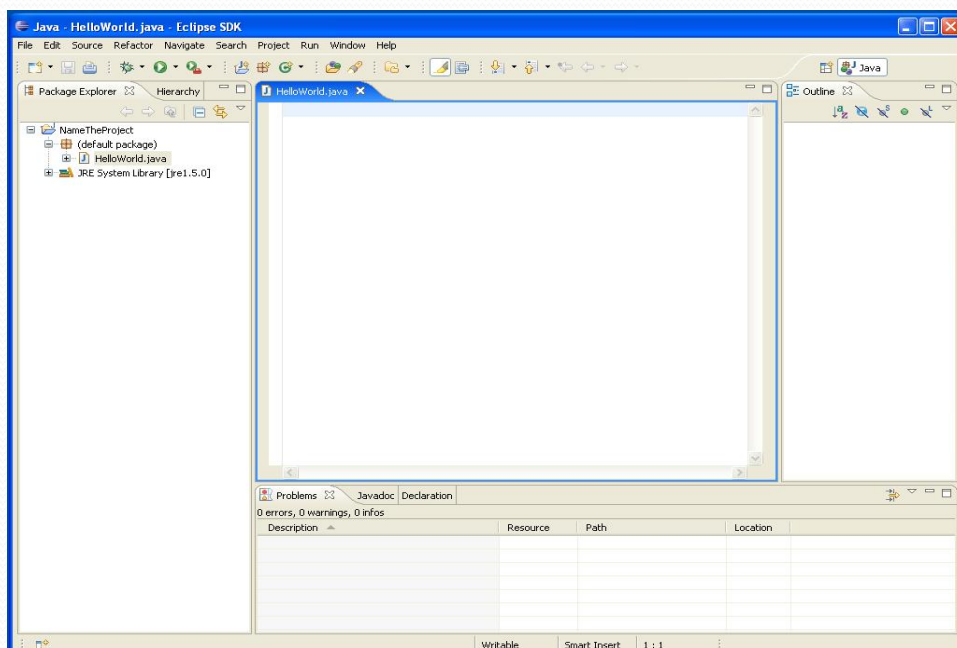
Step6: Now select the project in which you want to write. Then name the java class with extension (".java") and click "Finish".



3/3/10

11

Step7: Now you have the editor space, start coding.



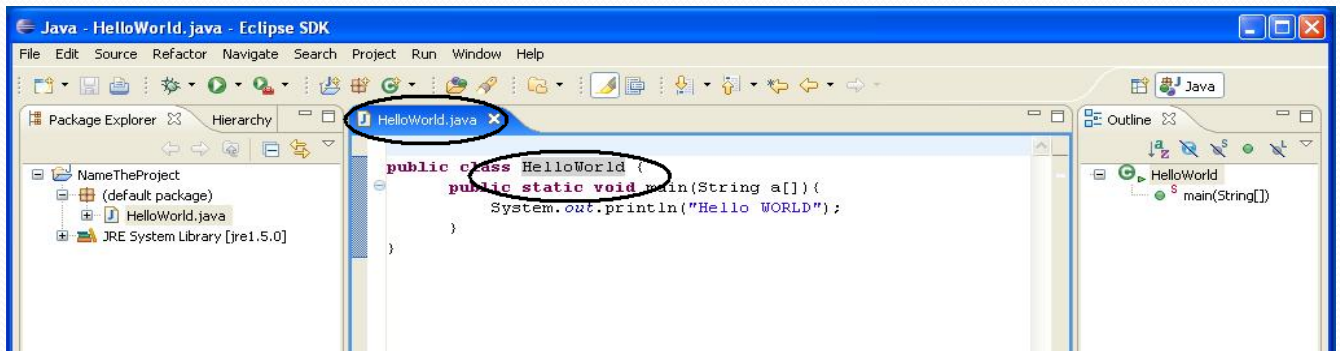
3/3/10

12

Writing the code

Step8: In eclipse when ever you save the file, it will compile the code by default.

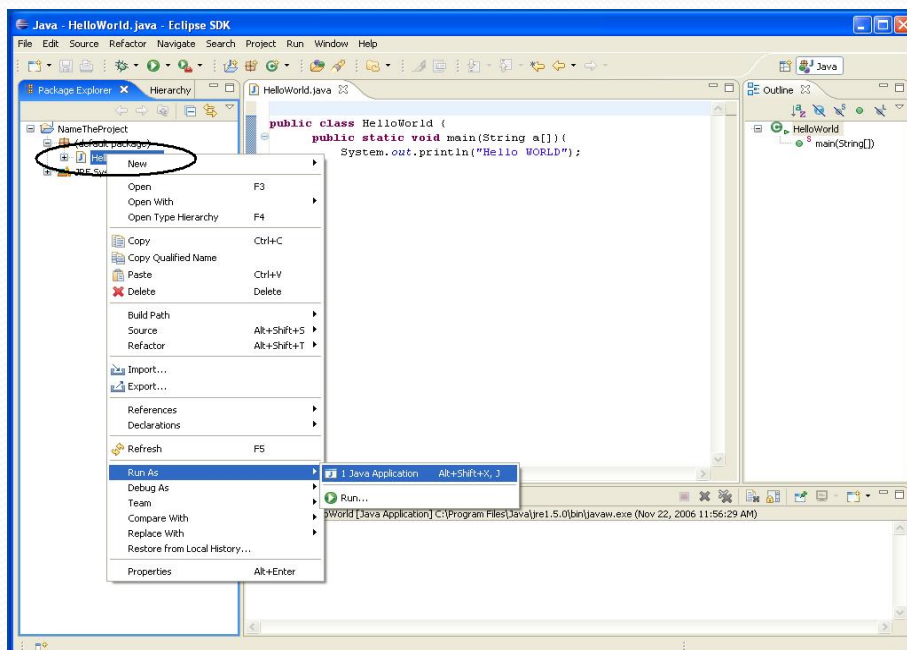
- Basic tip: Class name and the file name should be same.



3/3/10

13

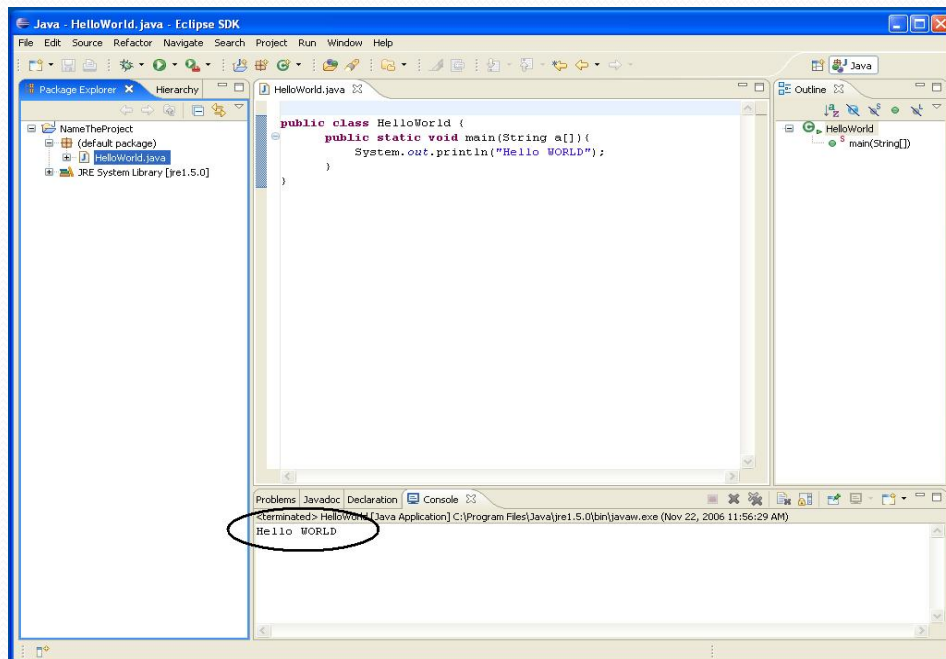
Step9: Running the java class. Right click on the class file and choose run.



3/3/10

14

Step10: Here you can find the output.

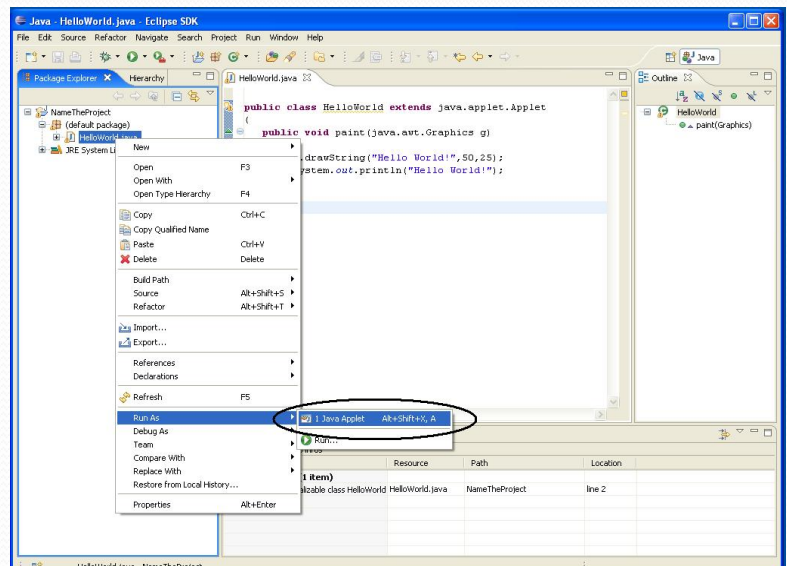


3/3/10

15

Running an applet is no different, it is the same.

Eclipse Identifies the program from the class and runs the file as an applet (by default). Try it out. The only option you get is the "Run as Applet"



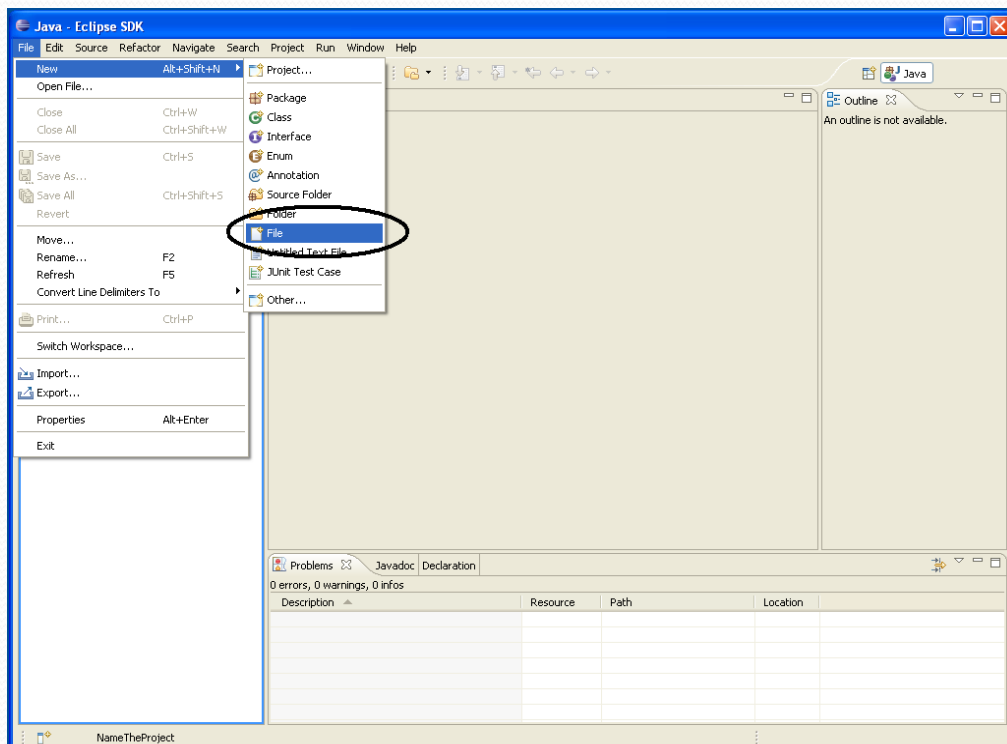
3/3/10

16

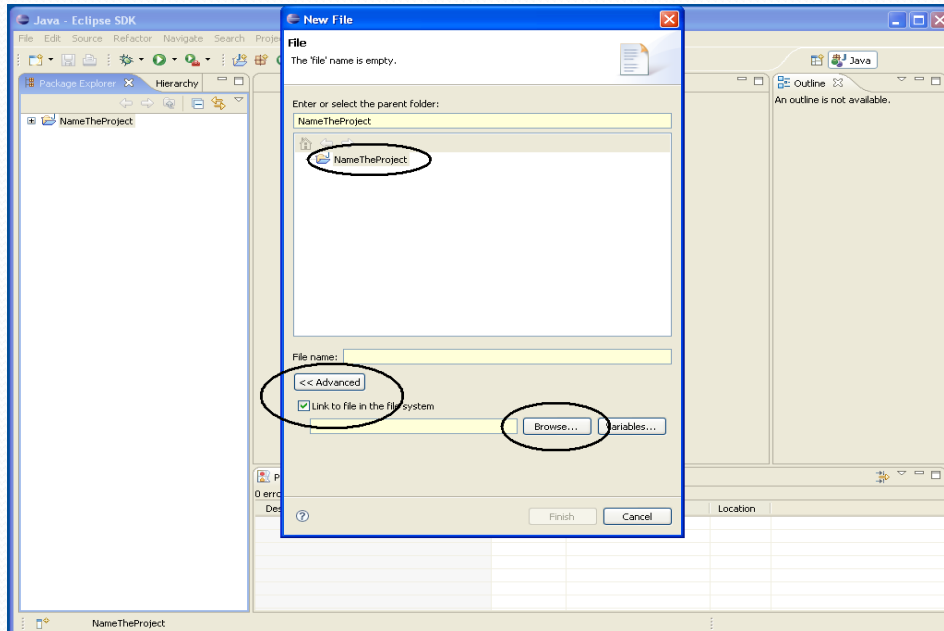
Importing a File into Eclipse

No need to code everytime.

Step1: Go to the “File” option , choose “New” and then “File” as shown in the figure below.



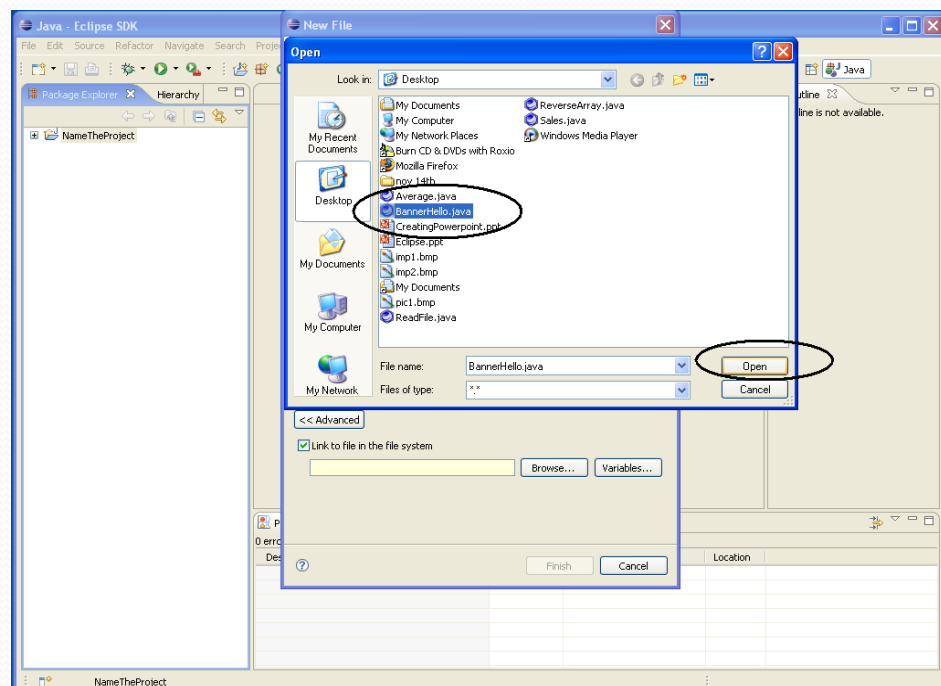
Step2: Select the project in which you want to import the file and then the advanced option, click the check box and browse.



3/3/10

19

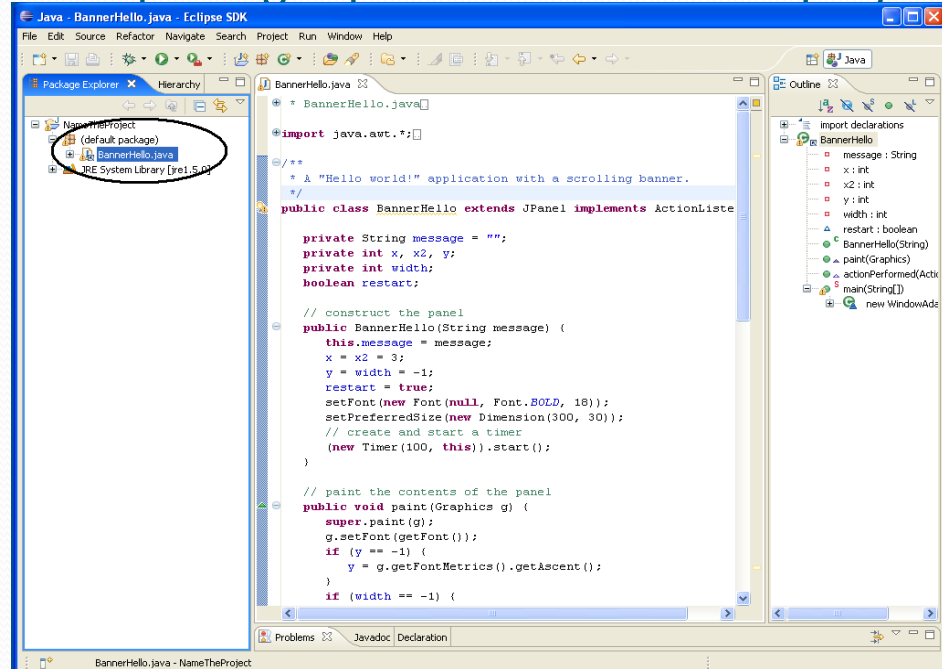
Step3: Select your file from the location and click the “Open” option.



3/3/10

20

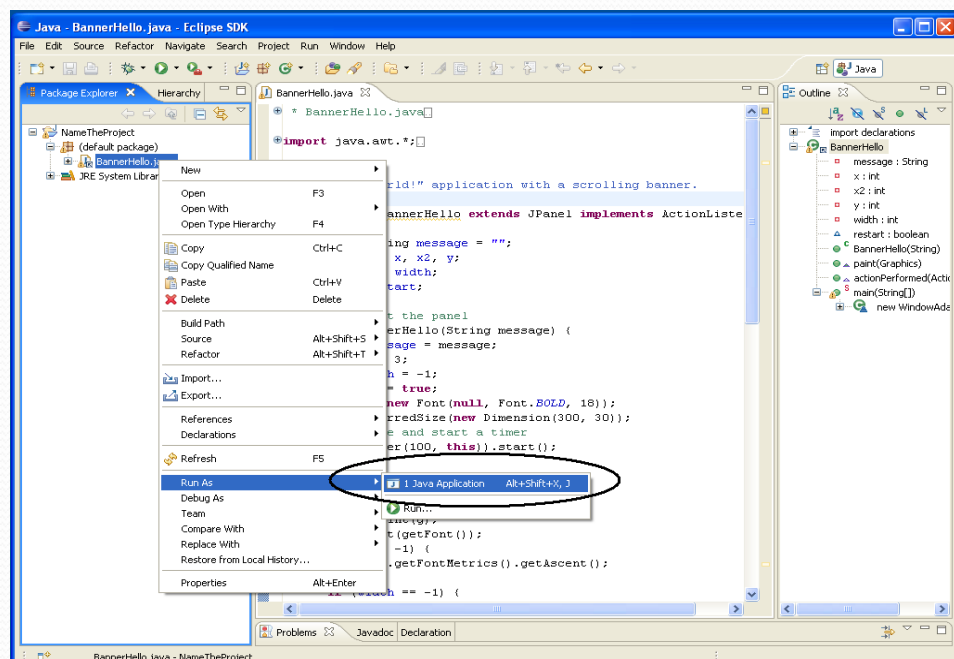
You can see that the imported file is shown in the work space and is loaded in the default package space of the selected project.



3/3/10

21

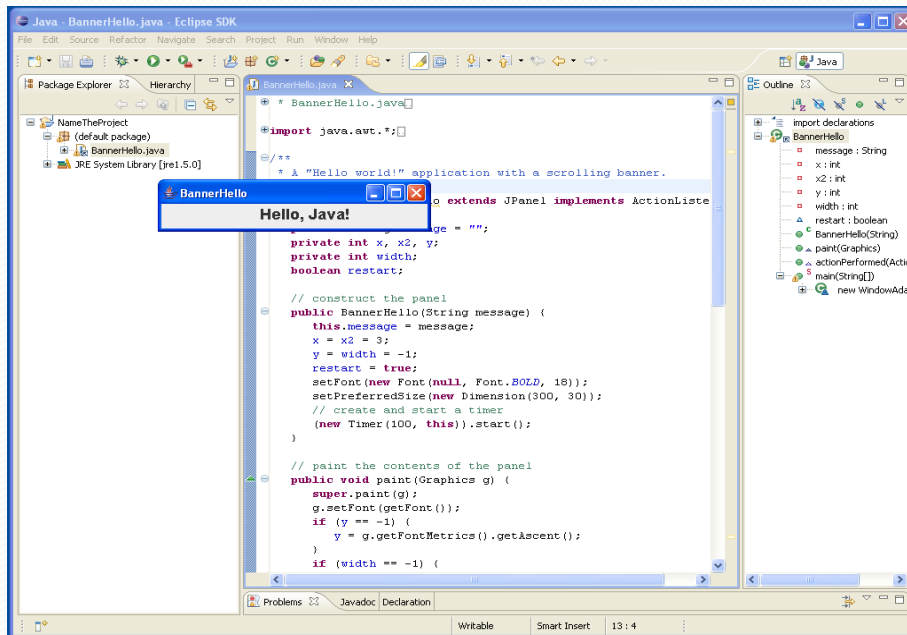
Step4: Now all you need to do is to select the file and run it. You can see the output.



3/3/10

22

Here's the output of the applet.



3/3/10

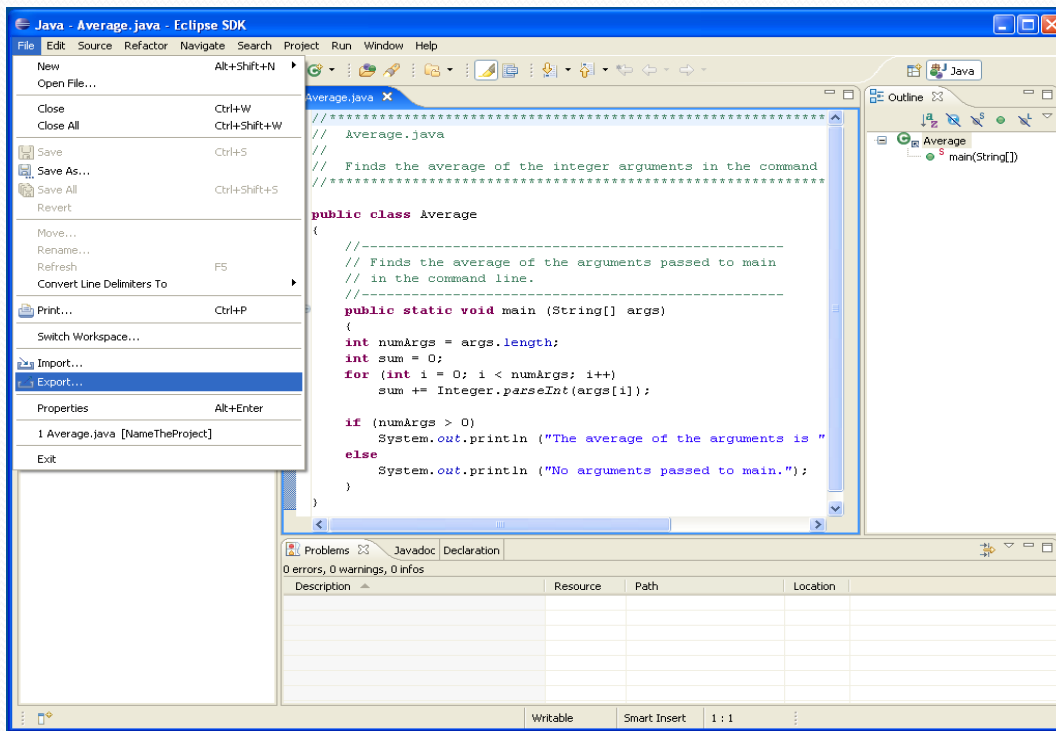
23

Creating a .jar file

Creating a .jar file is very useful especially when we are working on different systems every time.

Before going further one should know that to create a .jar file, we need to have that particular project present in the Eclipse.

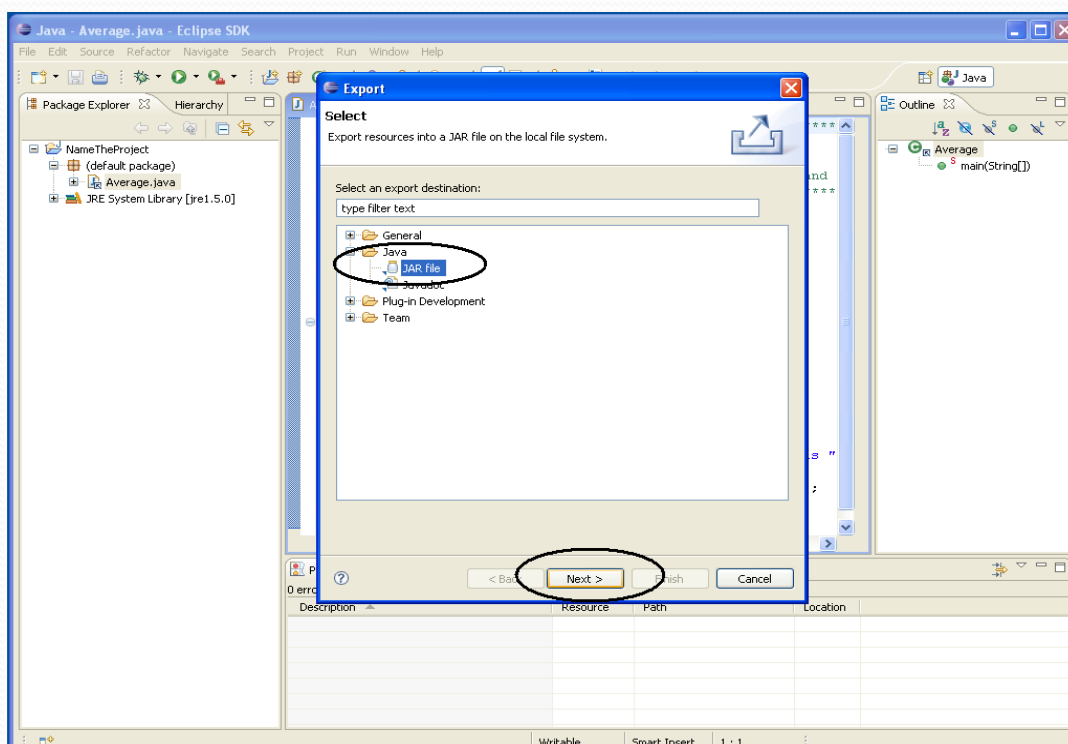
Step1: Select “File” and choose “export” option as shown below.



3/3/10

25

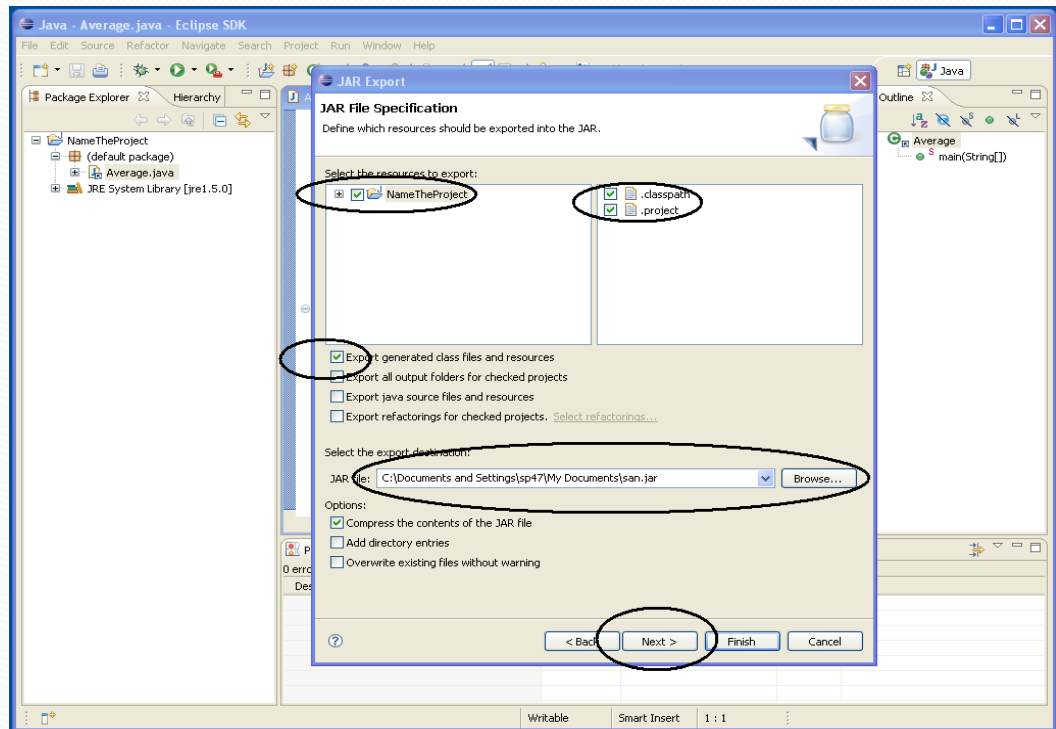
Step 2: Choose “jar file” and “next”.



3/3/10

26

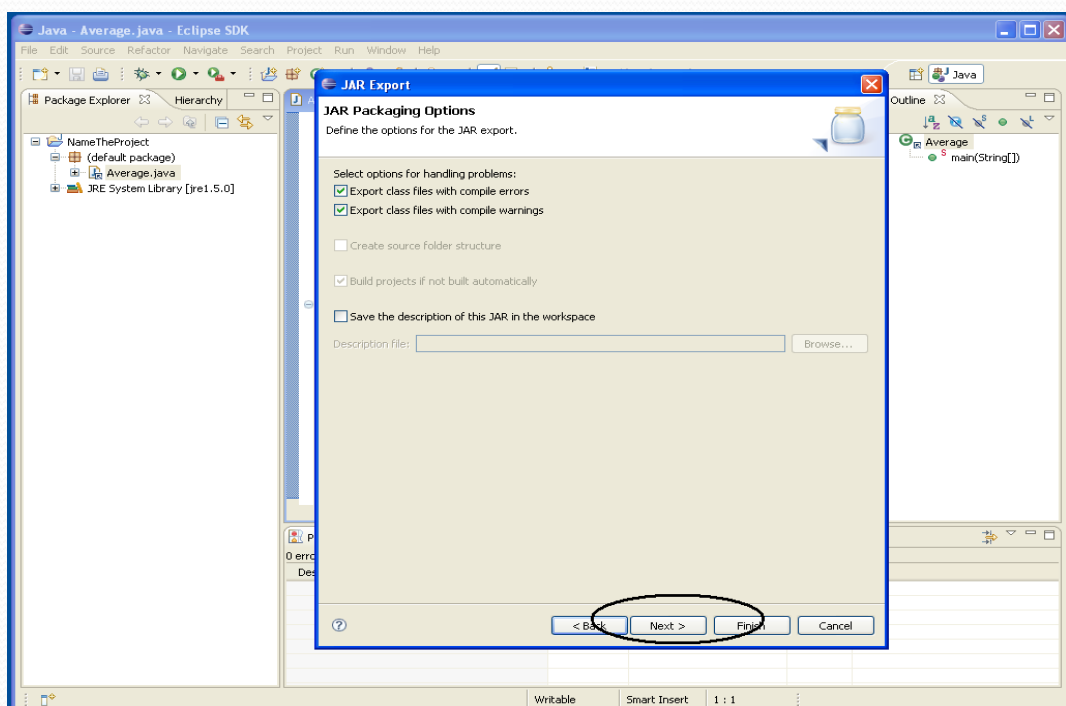
Step 3: Select the project and the files you want to add. Browse through and select your destination file.



3/3/10

27

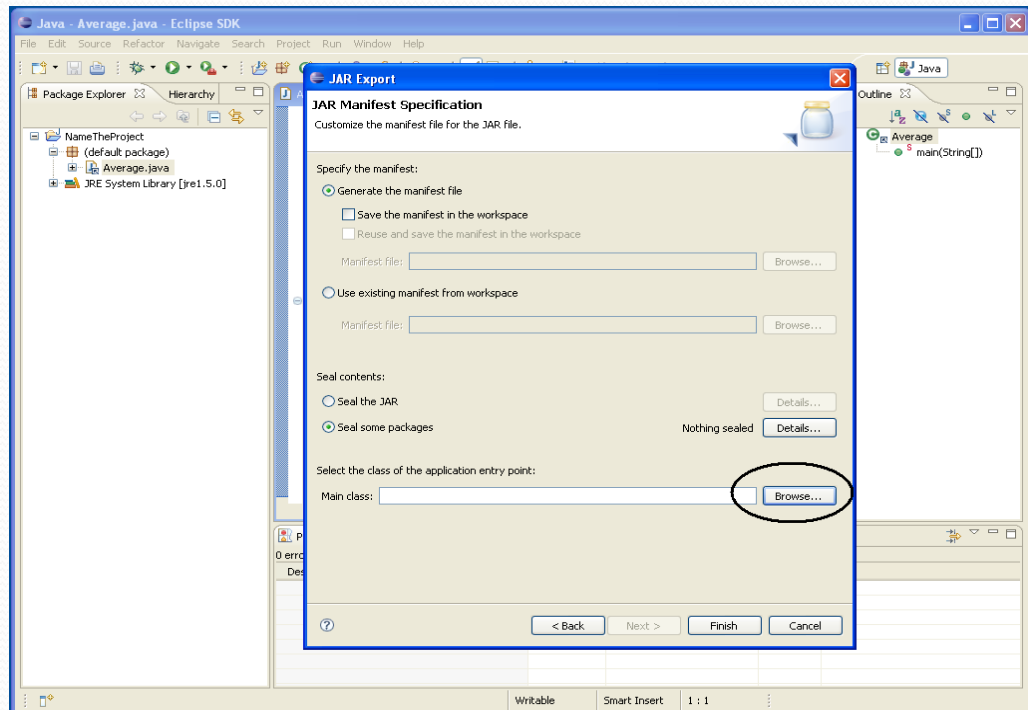
Step 4: In the Packaging options, the first 2 options are usually enough. You can choose the other advanced options depending on your requirements.



3/3/10

28

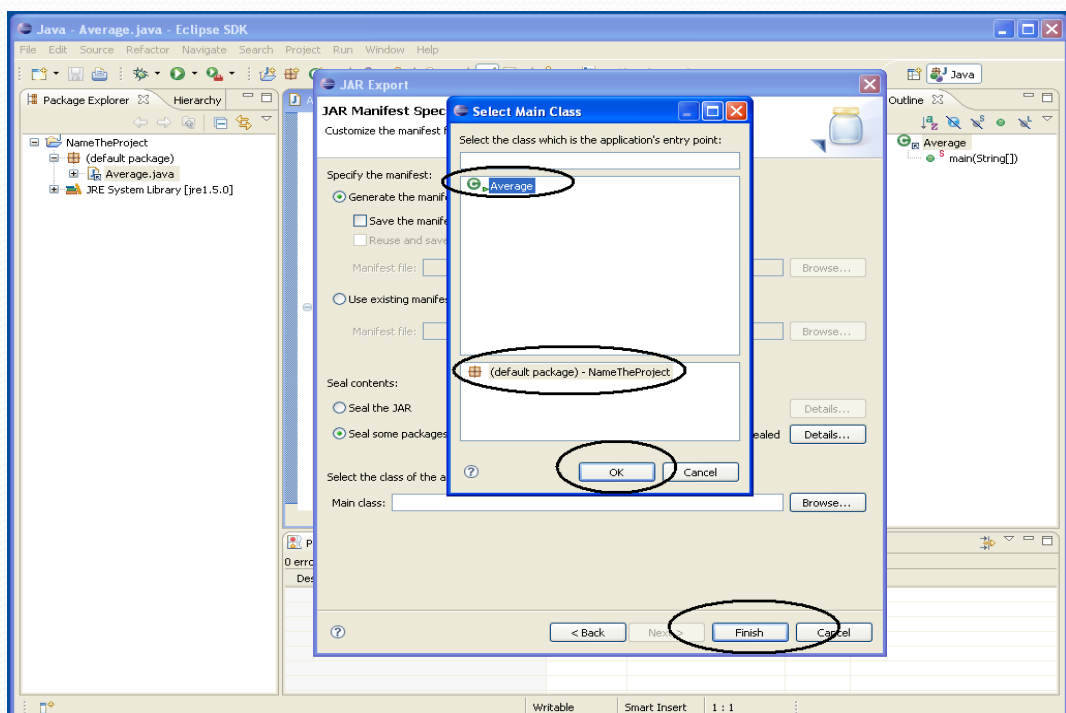
Step 5: Jar export. The options chosen here are the standard ones. We then browse through and select the class file. (Read carefully before selecting options).



3/3/10

29

Step 6: Eclipse by default shows the class files and the projects available. We just need to choose the right one and the jar file is created.



3/3/10

30

Some Information and FAQ on Eclipse

Will help starters a lot

Features of Eclipse

- Eclipse has the basic features required for editing, running, and debugging Java code, although they do some things slightly differently. In addition to basic programming features, Eclipse support for more advanced Java development tools such as Ant, CVS, JUnit, and refactoring.
- Often, the hardest thing about migrating to Eclipse is learning how to do old things in the new environment. But Eclipse has a complete and easy-to-use help system with online documentation.
- Eclipse's GUI builder is a separate component.

Adding Eclipse plugin

The simplest way is to copy the plugin's folder to the plugins subfolder of the appropriate Eclipse binary folder. This method however, requires that separate copies of plugin binaries be created for different platforms.

Running a code

- Eclipse uses an incremental compiler, so it isn't necessary to explicitly compile your Java files; the compiled class files are saved automatically when you save your Java files.
- To run a program, the easiest way is to select the file containing a `main()` method in the Package Explorer and then select **Run > Run As > Java Application** from the main Eclipse menu.

Debugging

- First, set a breakpoint in the `main()` method by double-clicking in the left margin next to the call. If this code were a little less trivial, it would also be possible to set a conditional breakpoint -- one that stops when a particular expression is true, or one that stops after a specific number of hits -- by right-clicking the breakpoint and selecting **Breakpoint properties** from the context menu.
- To start debugging, select **Run > Debug As > Java Application** from the main menu. Because Eclipse has a Debug perspective that is better suited for debugging than the Java perspective, it will ask if you want to change to this perspective.

The famous Don't:

- Set the work space of the eclipse where you can easily access it.
- Never start writing the code without making a project. You need to create a project folder every time you start a new assignment
- Main classname and the file name should always match.

Strange Error:

- The Eclipse IDE reports a strange error similar to: "Cannot create workbench".
- In many situations this problem can be resolved by deleting the workspace / .metadata / .registry file in user's home directory and restarting the IDE.