


# 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](http://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"
- It is an **Integrated Development Environment (IDE)**, that allows to manage the whole development process of Java applications, by providing many features for programming (editor, debugger, etc.)
- It supports other languages by means of plug-ins (C/C++)
- Multi-platform (Linux, Windows, Mac OS)

# What's Eclipse?

- 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.

# Getting Eclipse

- In the Lab: already installed.
- On your laptop
  - You will need to install a Java Virtual Machine (JVM)
    - <http://www.oracle.com/technetwork/java/javase/downloads/index.html> (JDK)
  - Download the latest version at:
    - <https://eclipse.org/downloads/> (Eclipse IDE for Java Developers)
    - Decompress the downloaded package and click **eclipse.exe** (under Windows) or run **eclipse** (under Linux)
  - Installation steps at <http://wiki.eclipse.org/Eclipse/Installation>

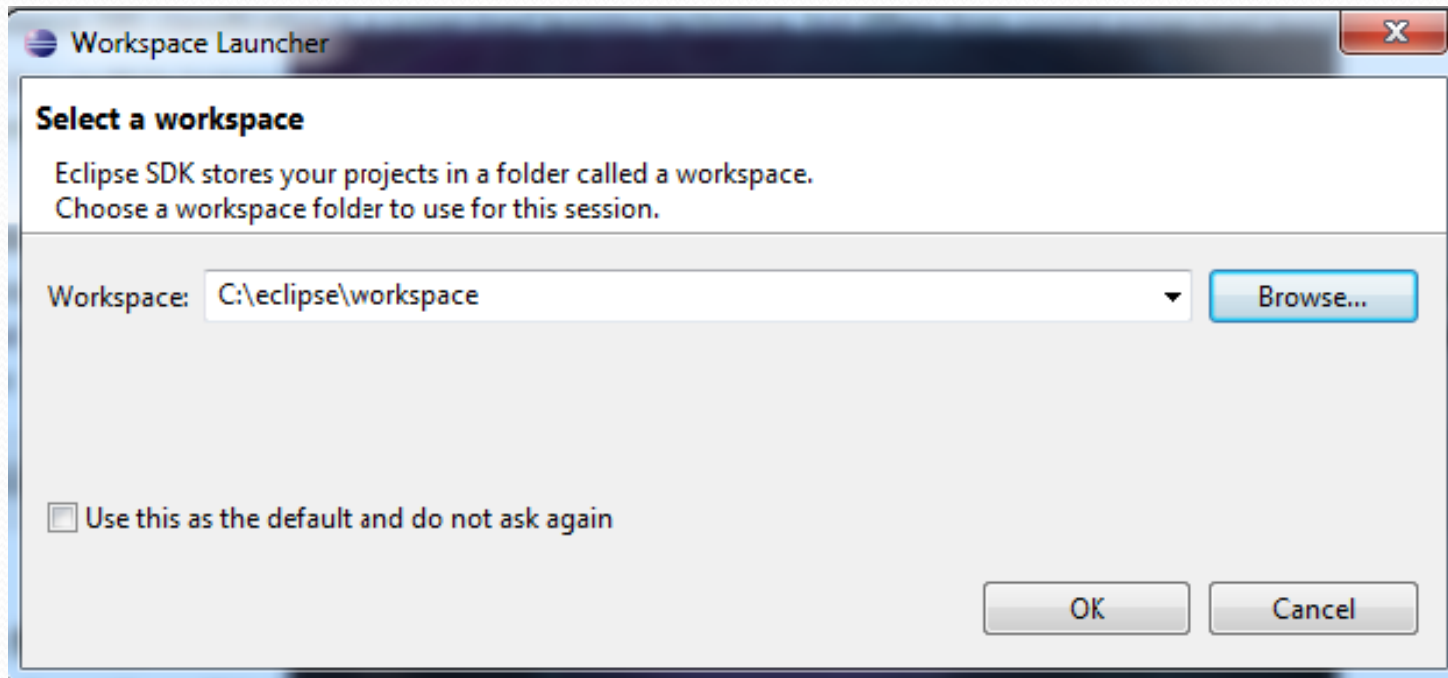
# Essentials of Eclipse

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

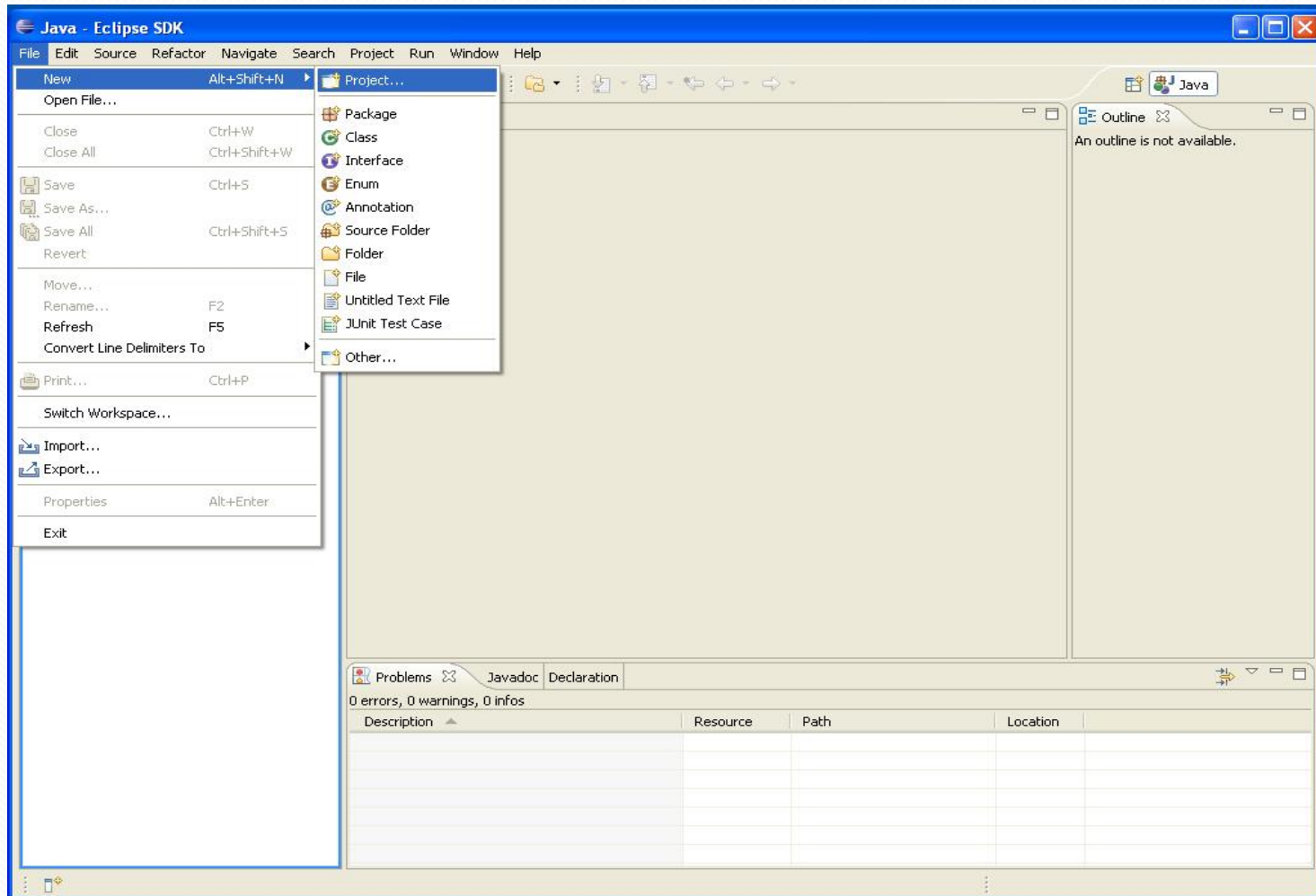
- In Eclipse you need to start of with **creating a project**.
- Then choosing a particular project, we **create** our **java file** in it. No need to worry, it is simple and you will learn it by the end of this presentation.
- In Eclipse you need not remember the exact command syntax, it helps you writing 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 Eclipse from start on your system, choose your workspace: workspace is the directory where the projects will be stored.

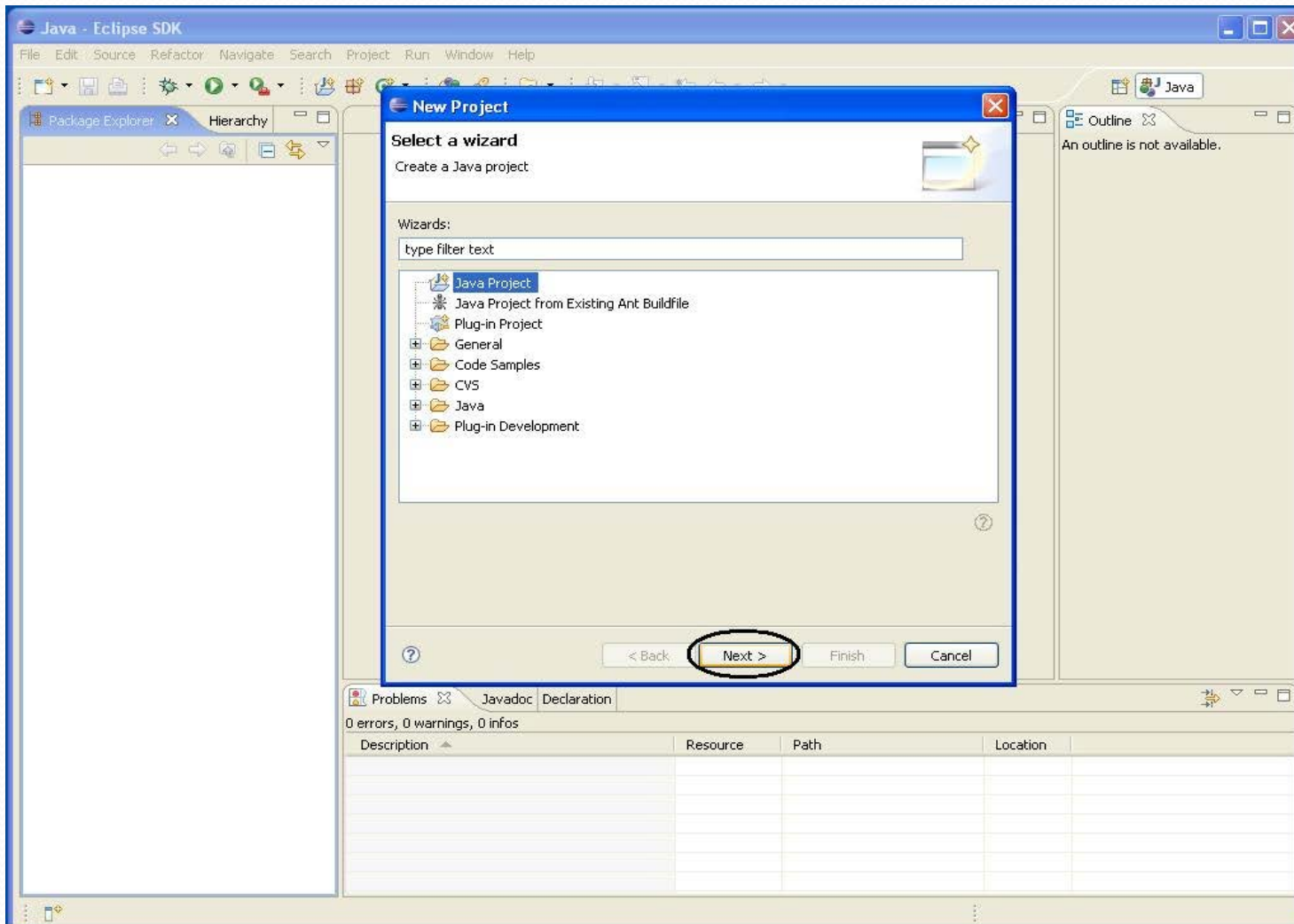


Step2: In Eclipse when ever you want to create a class, you need to select the new project by default (File → New → Project).



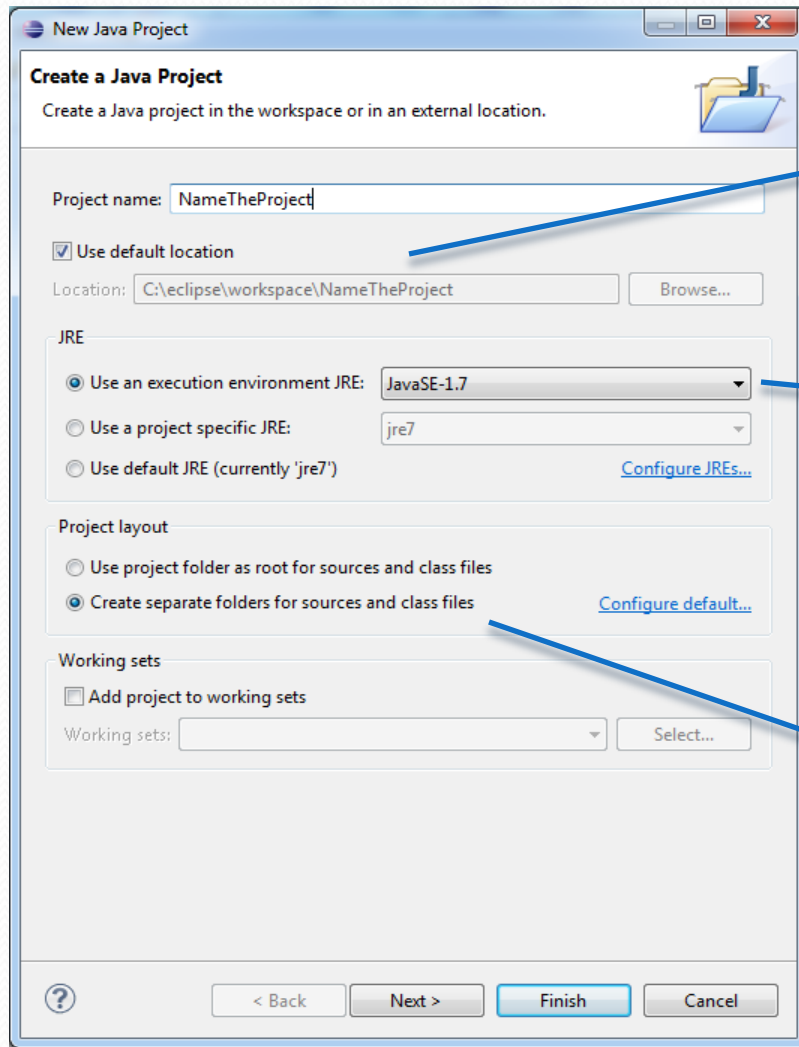


Step3: Select *Java project* and click next.



You can also choose *File* → *New* → *Java project* directly.

## Step4: Name the project and click finish.



Creates the directory with the specified «project name» in the workspace path.

Specifies an execution environment to be used for the new project (Java Runtime Environment).

Creates a source folder (*src*) for Java source files and an output folder which holds the class files of the project. When selecting the other option, the project folder is used both as source folder and as output folder for class files.

# Step5: IDE views

The screenshot shows the Eclipse IDE interface with the following components and callouts:

- Package Explorer:** Located on the left, it shows a project named "NameTheProject" with a "src" folder and "JRE System Library [JavaSE-1.7]".
- Views selector:** A toolbar icon (a window with a plus sign) used to show or hide views.
- Perspective:** A callout box explaining that it allows switching between different views like "Java", "Java Browsing", "Type Hierarchy", and "Debug".
- Editor:** The central area where source code is edited.
- Outline:** A callout box explaining that it displays an outline of the structure of the currently-active Java file in the editor area.
- Miscellaneous Views:** A callout box pointing to the bottom-right area, which includes a table for "Problems", "Javadoc", and "Declaration".

**Package Explorer**  
List of projects/files (source and class files, libraries, packages). The *src* folder is automatically created if the corresponding option was set in the previous step.

**Perspective**  
To switch among different perspectives (*Java*: the perspective shown; *Java Browsing*: to browse the project structure; *Type Hierarchy*: packages, types, members; *Debug*: for debugging a program (breakpoints, console, tasks, etc.))


**Views selector**  
Window → Show view

**Editor**  
Where the source code is edited.

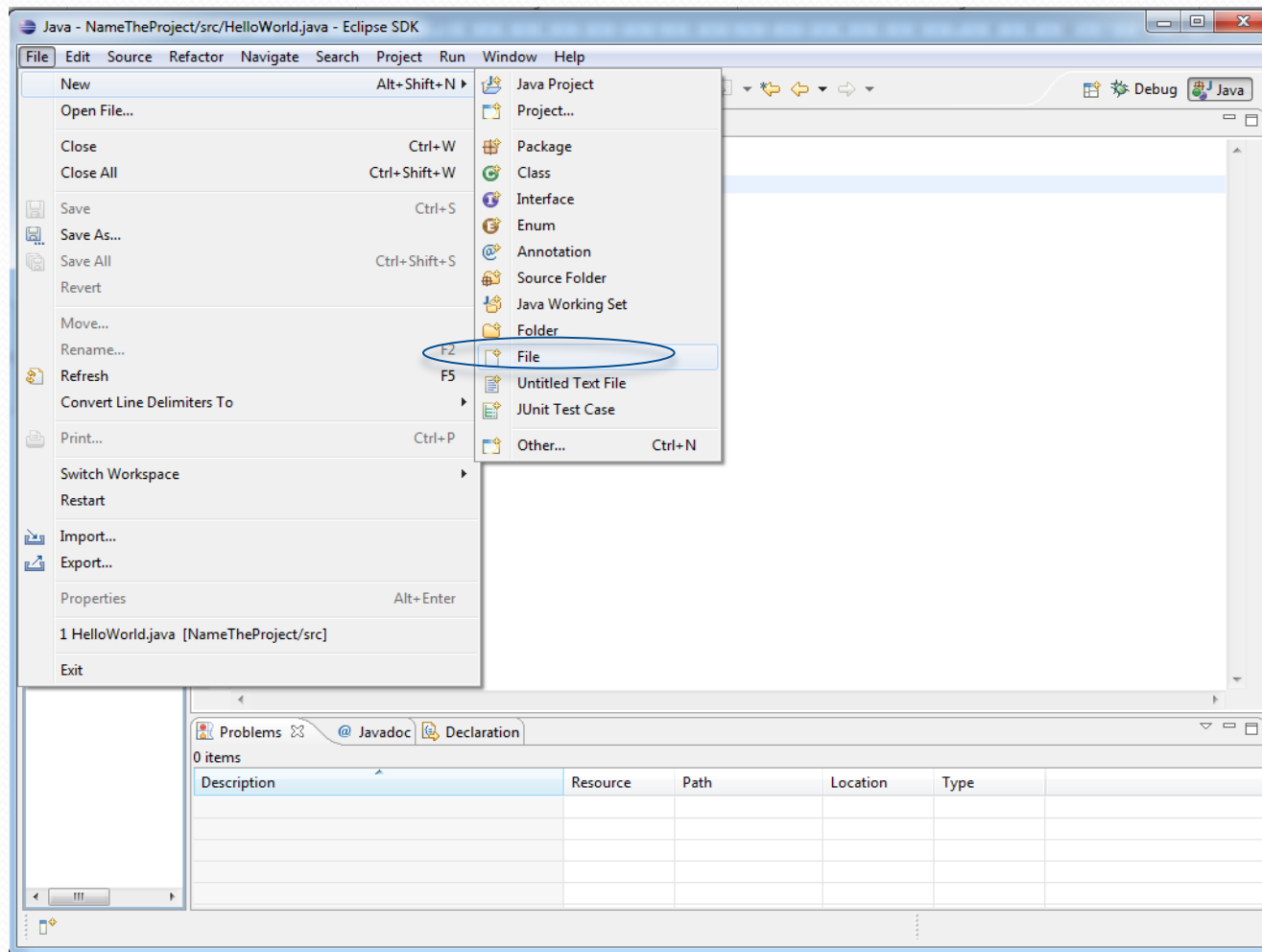
**Outline**  
displays an outline of the structure of the currently-active Java file in the editor area.

**Miscellaneous Views**  
Compiler problems  
Javadoc of the element selected in the Java editor  
Declaration: source of the element selected in the Java editor  
Console

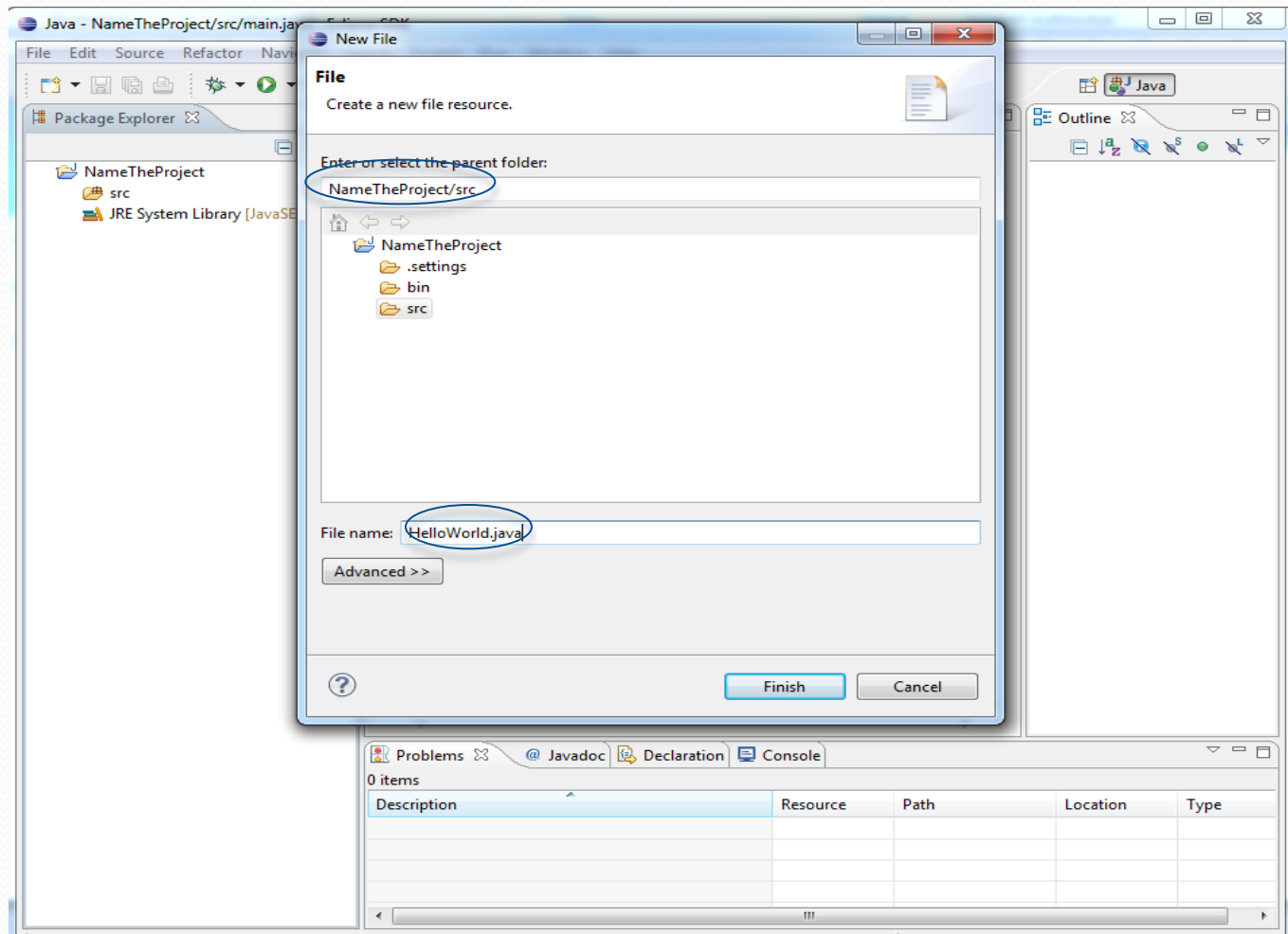
Description	Resource	Path	Location	Type
0 items				

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

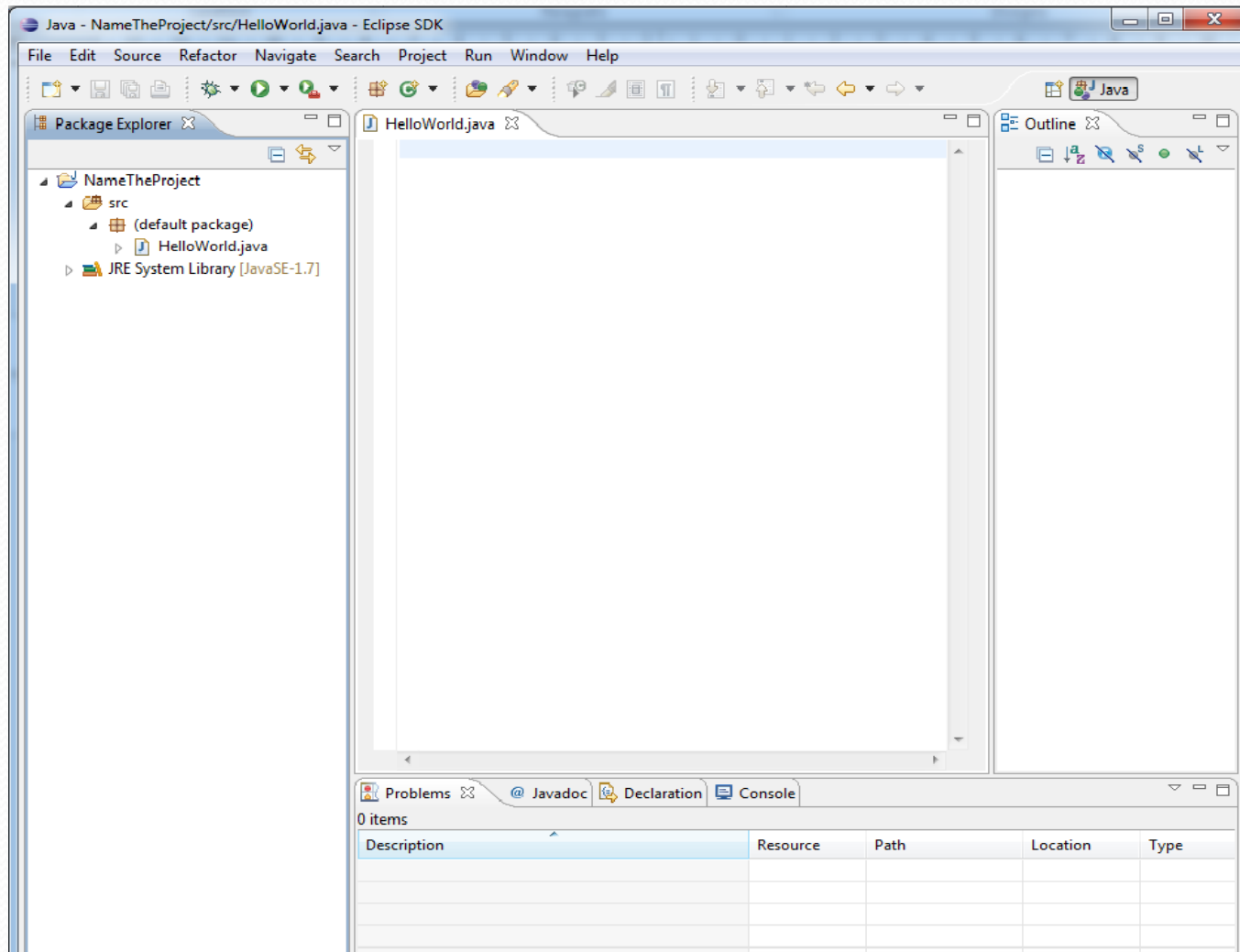
Step6: Now we create the java file by selecting the “File” menu, then “New” and “File”.



Step7: Now select the project in which you want to write, and the *src* folder if present. Then name the java class with extension (“.java”) and click “Finish”.



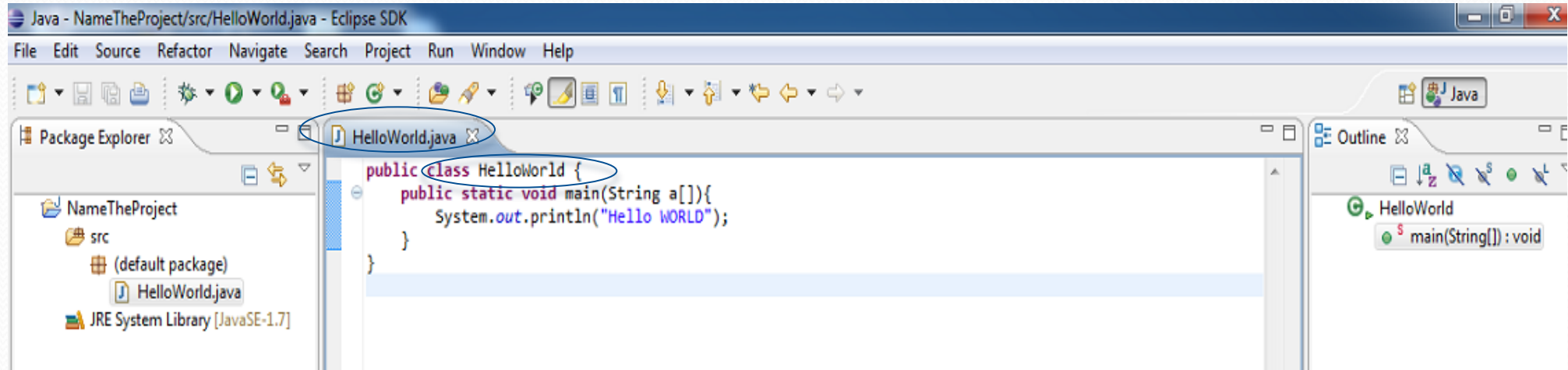
Step8: Now you have the editor space, start coding.



# Writing the code

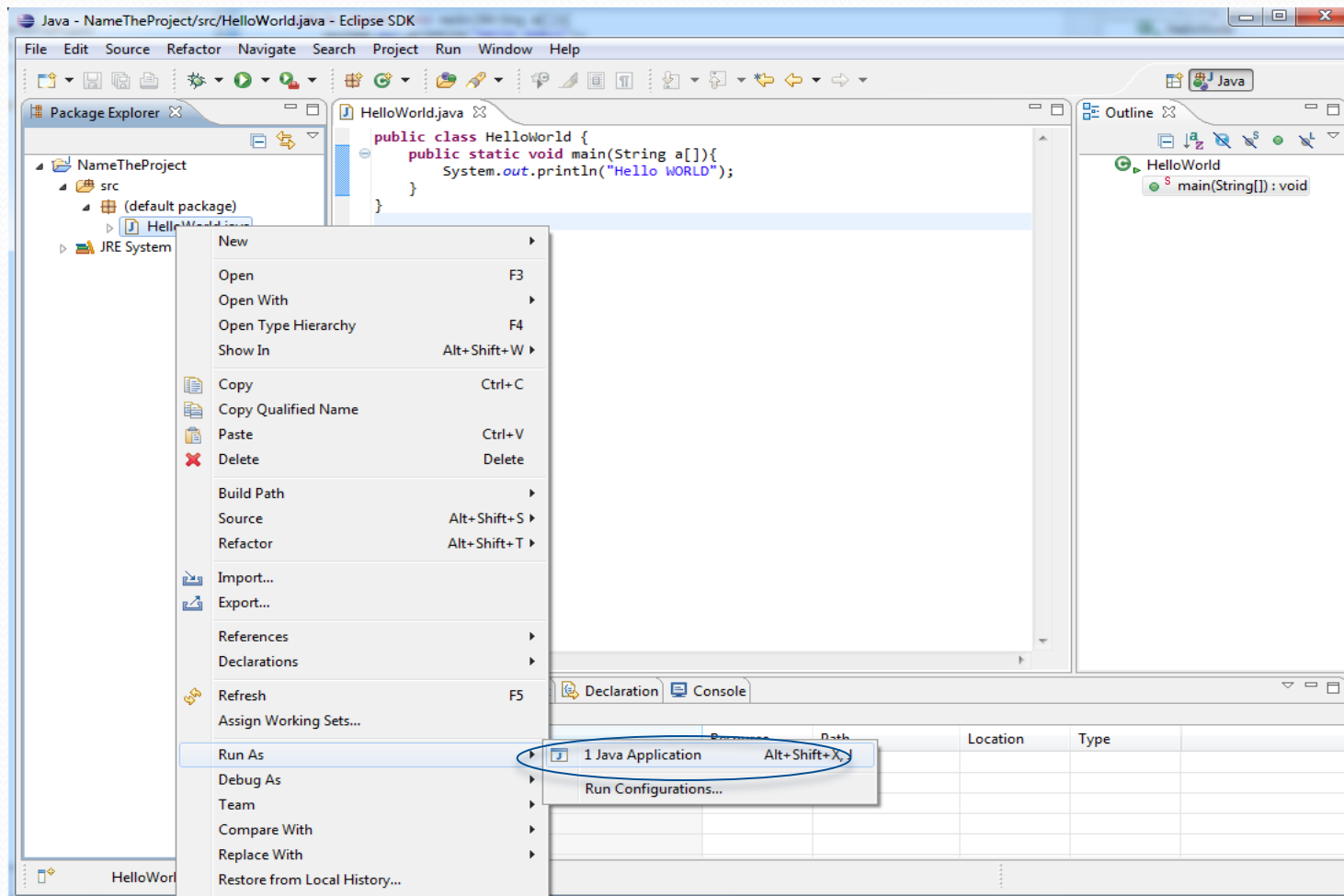
Step9: In Eclipse when ever you save the file, it will compile the code by default.

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

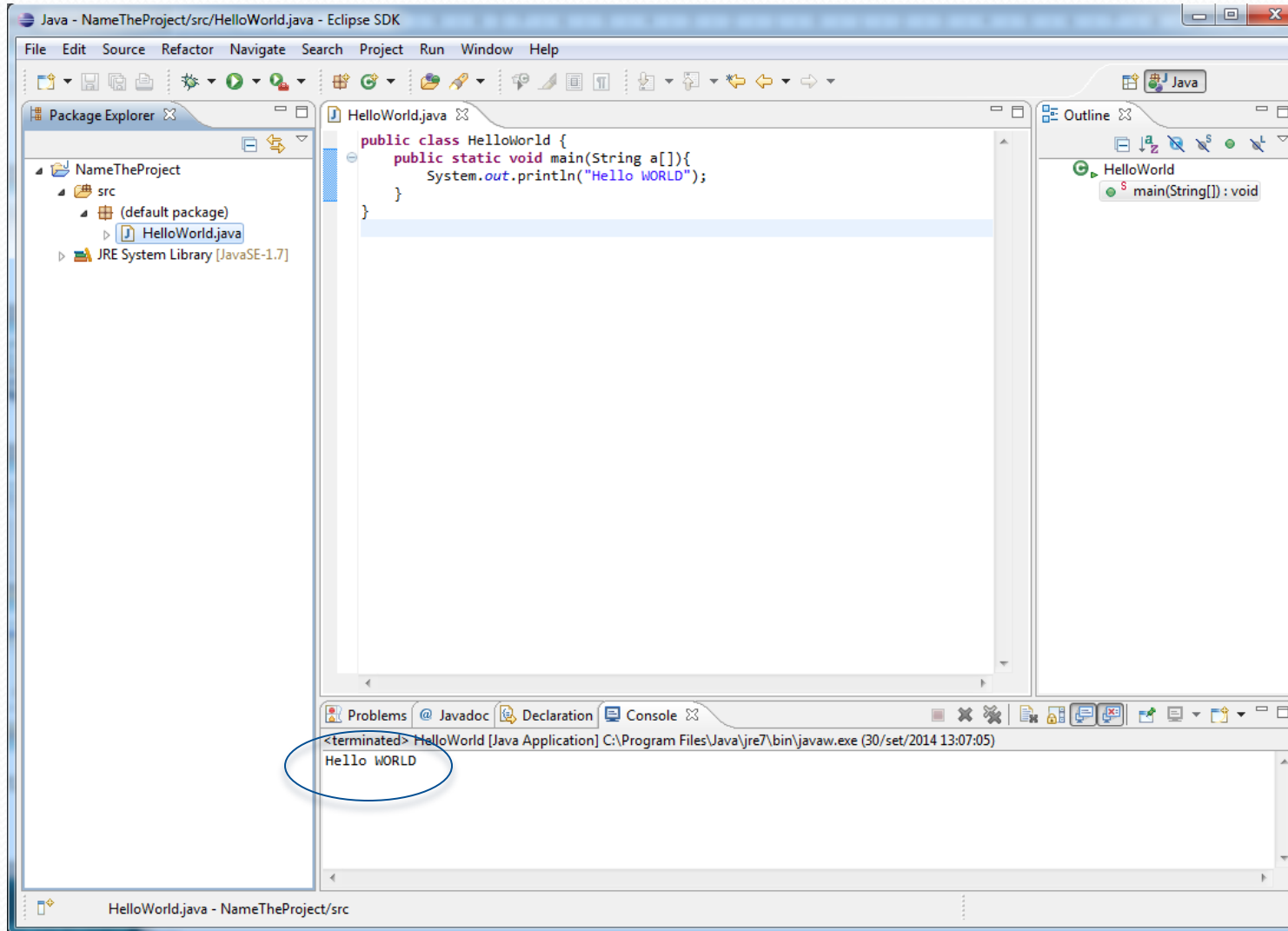




Step10: Running the java class. Right click on the class file and choose “Run as Java Application”.



# Step11: Here you can find the output (Console).



# Using the command line

- Type the Java program using an editor (vi/emacs on Linux, Notepad on Windows, etc.) and save it with the *.java* extension
- Compile the program with the Java compiler:
  - `javac HelloWorld.java`
  - The compiler produces a `.class` file called `HelloWorld.class`. It translates the Java source code into bytecodes for the Java Virtual Machine (JVM)—a part of the JDK.

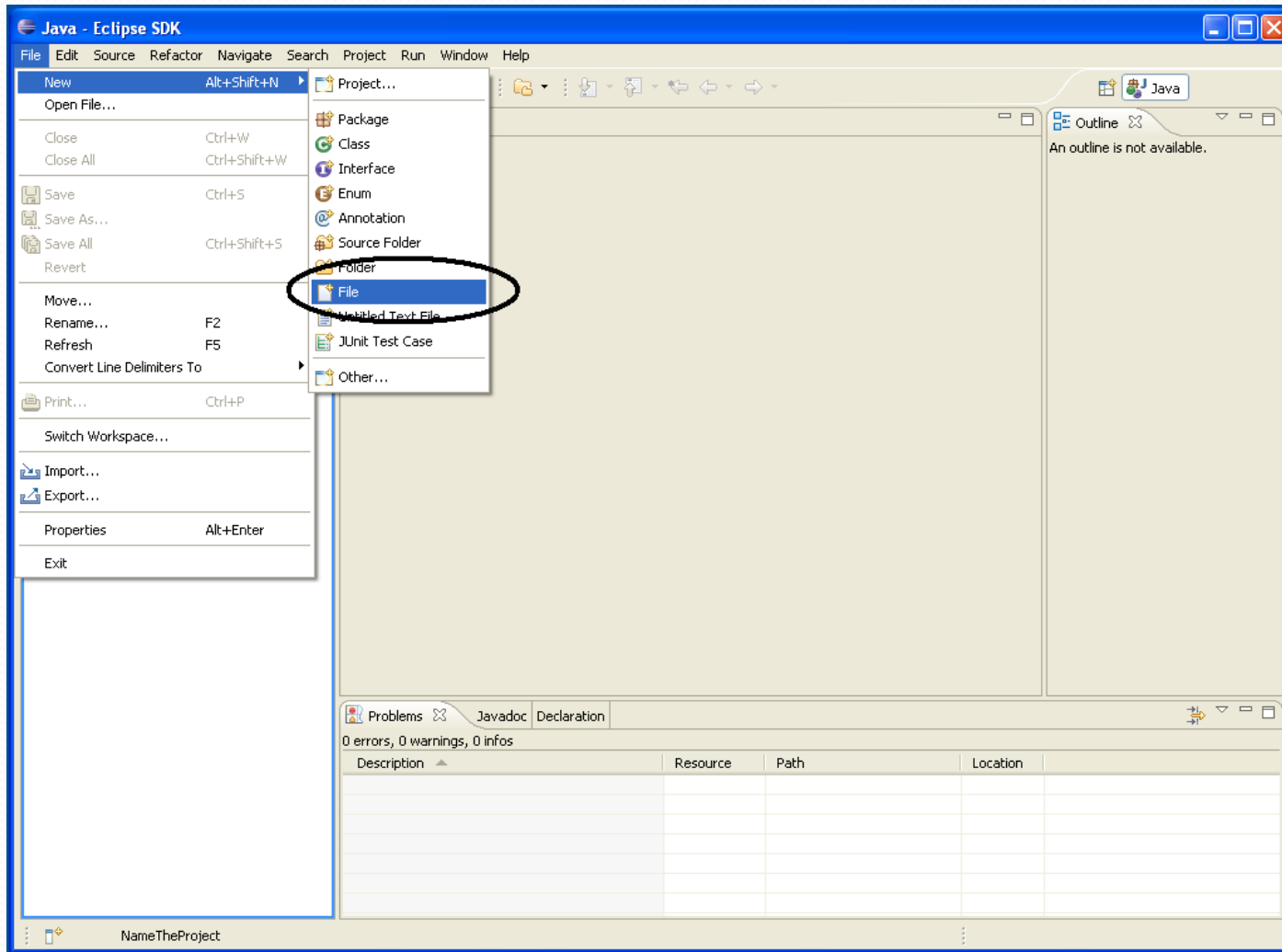
# Using the command line

- The JVM is invoked by the **java** command. For example, to execute the example Java application type:
  - `java HelloWorld`
  - The output will be shown in the command window.
  - In general (more than one source file), specify the name of the class which is the application's entry point (contains the `main()` method)

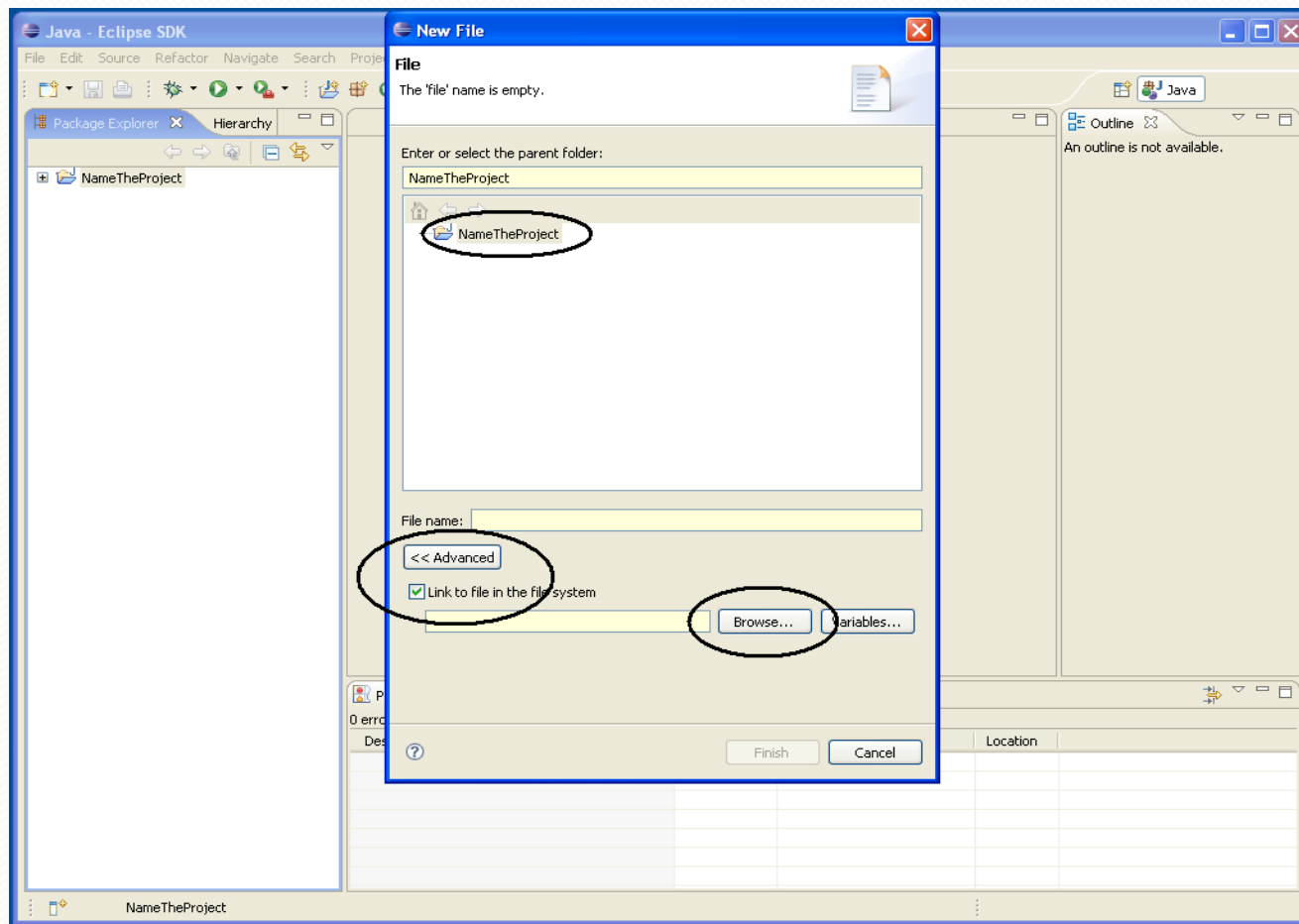
# Importing a File into Eclipse

No need to code everytime.

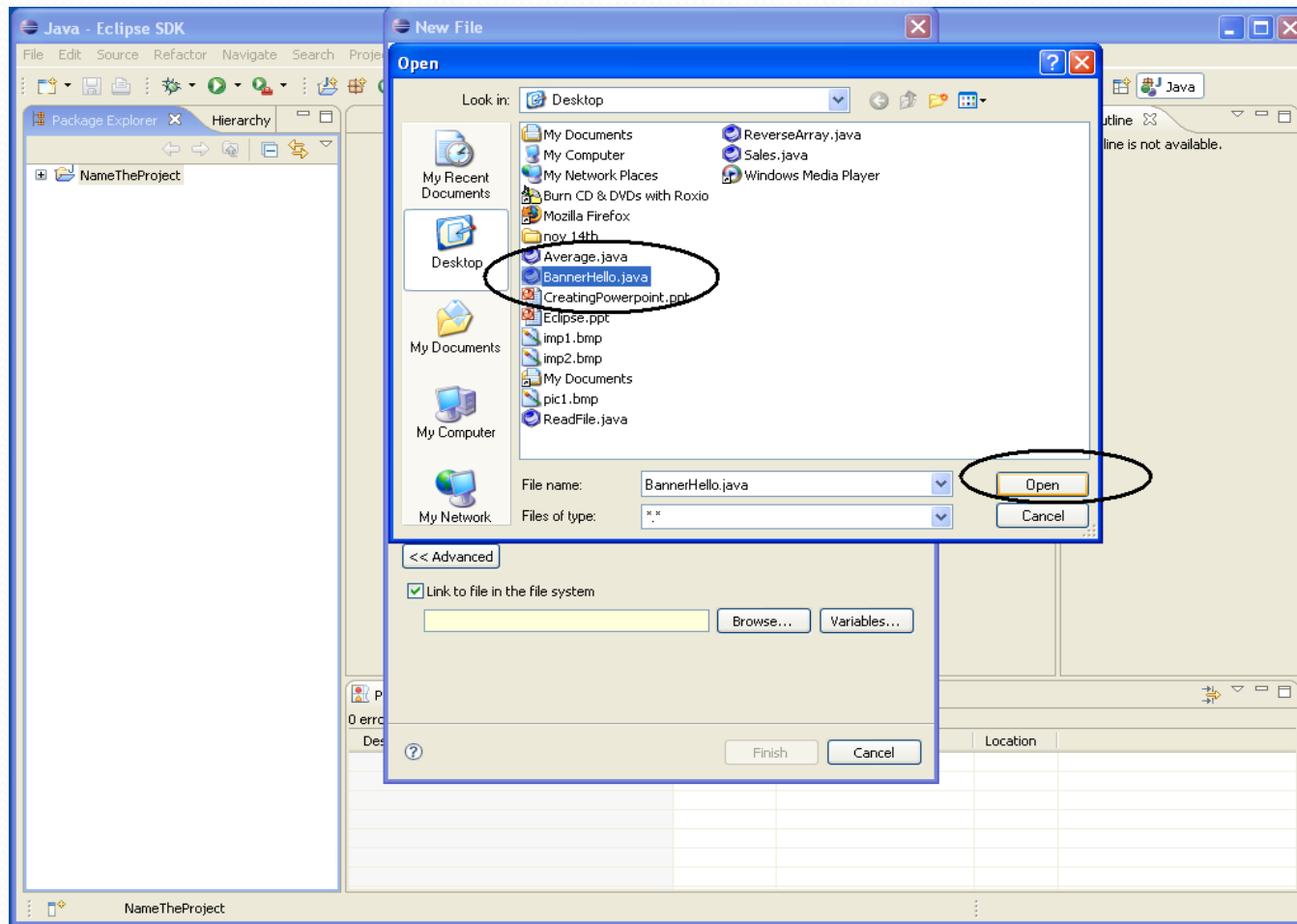
Step1: Go to the “File” option , choose “New” and then “File” as shown in the figure below.



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

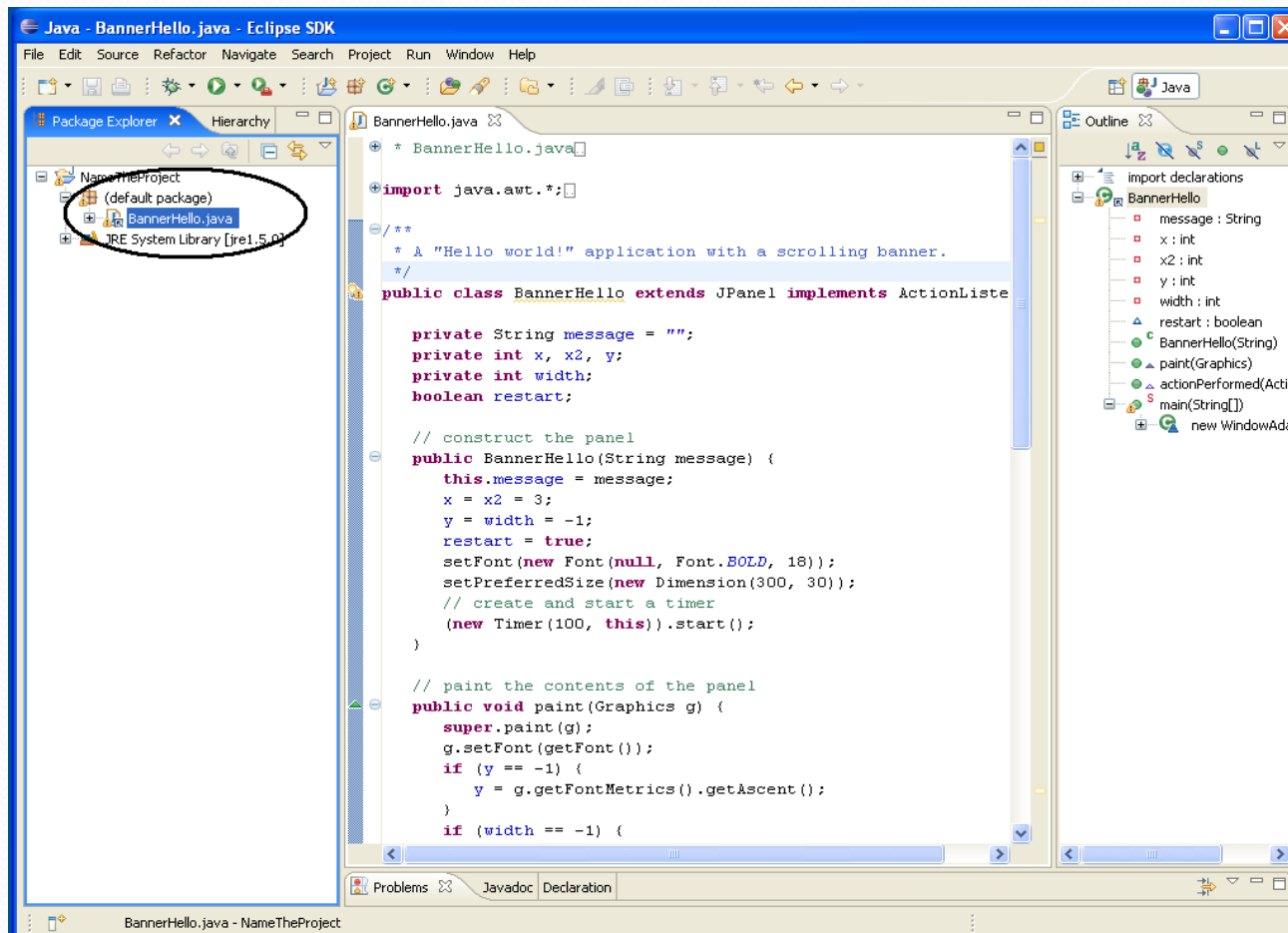


Step3: Select your file from the location and click the “Open” option.

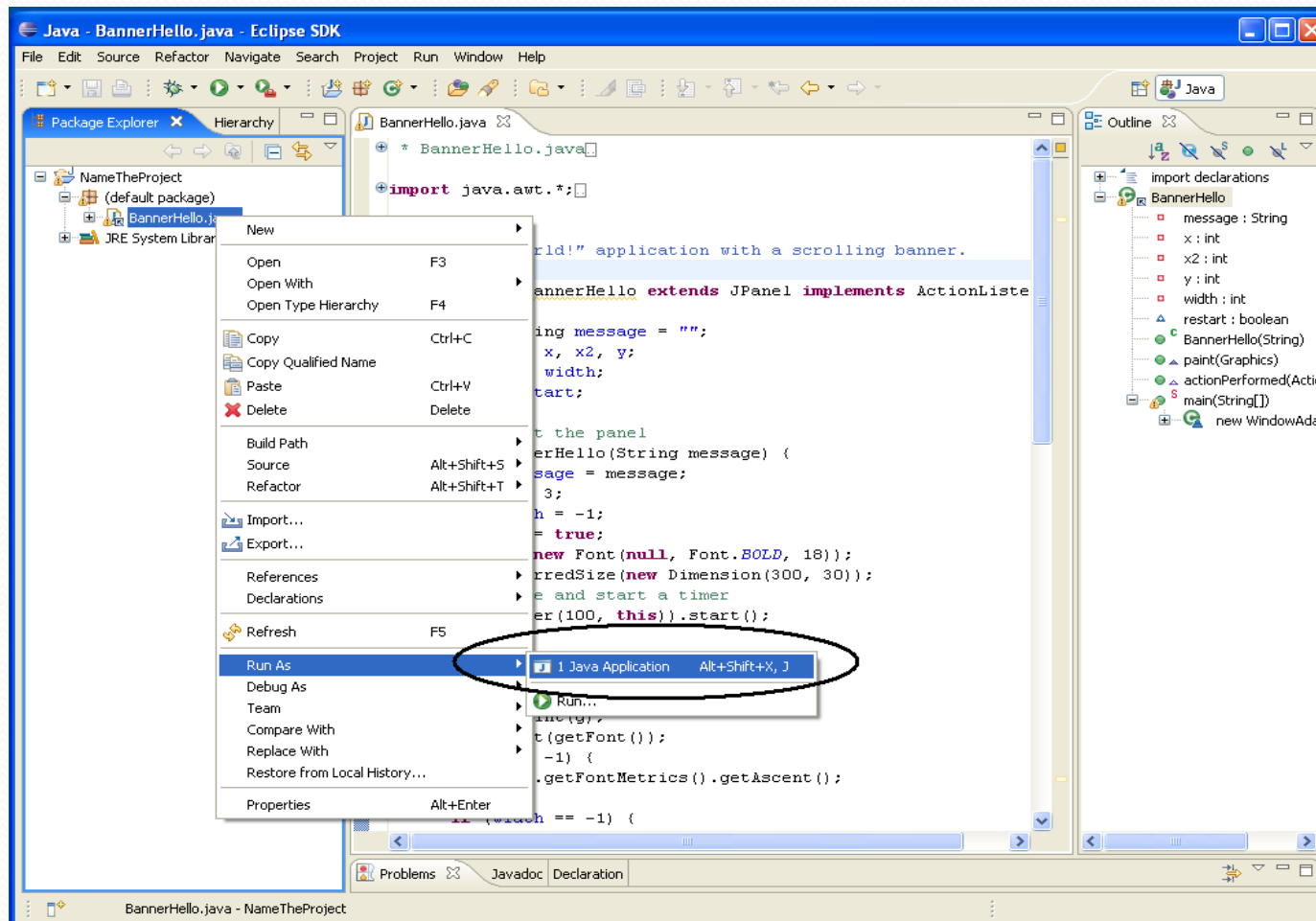




You can see that the imported file is shown in the workspace and is loaded in the default package space of the selected project.



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

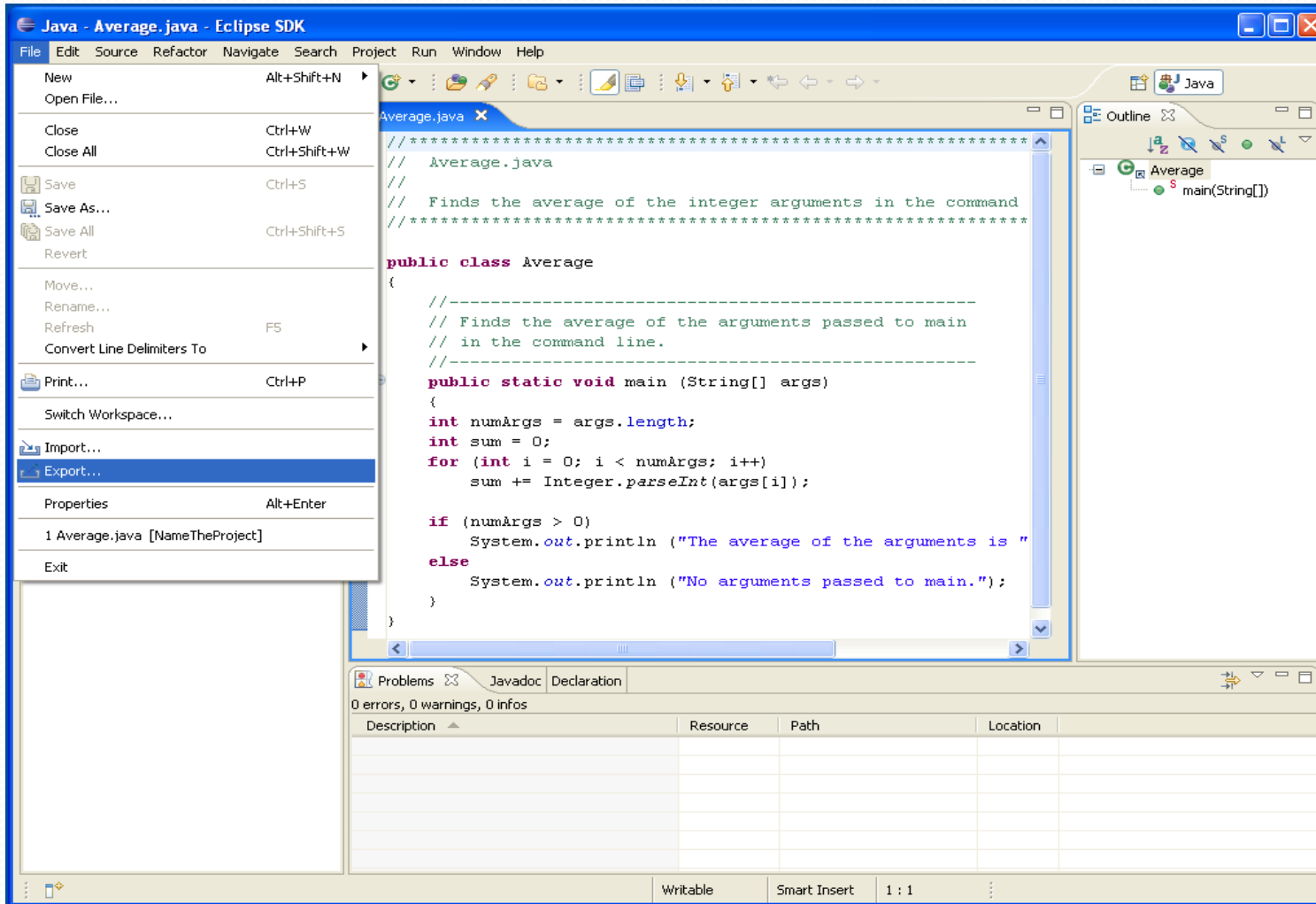


# Creating a .jar file

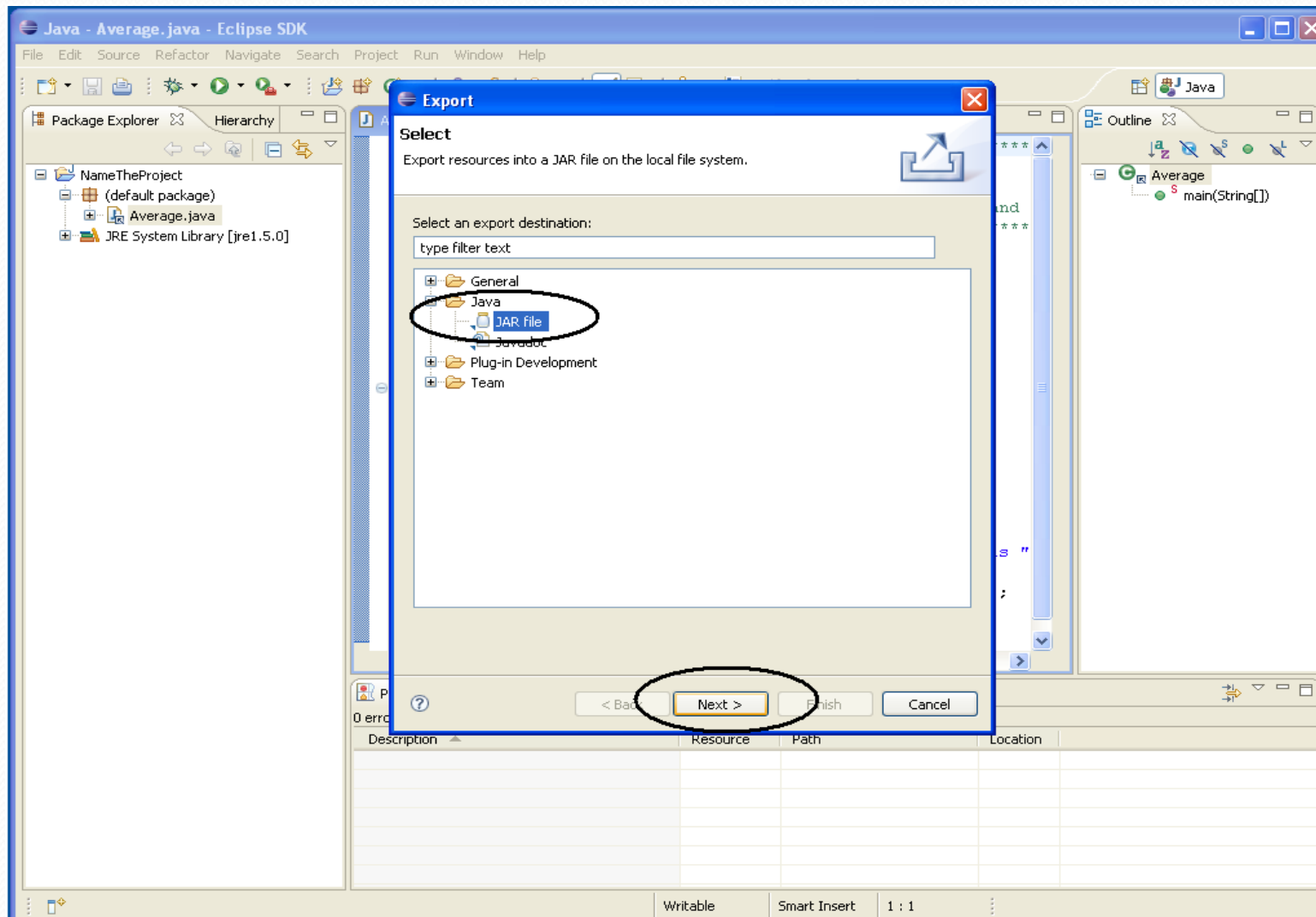
Creating a .jar file is very useful especially when we are working on different systems every time.

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

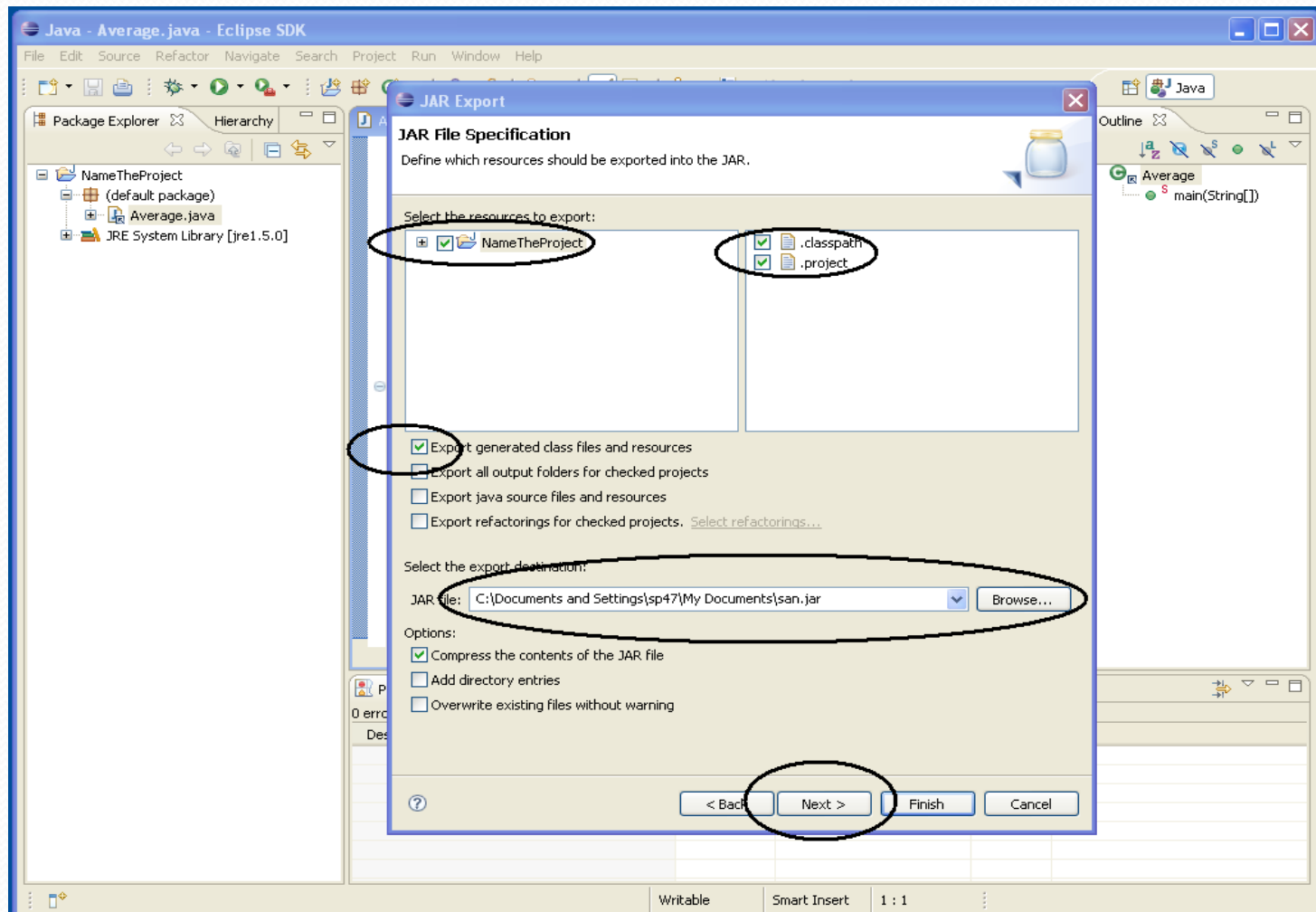
Step1: Select “File” and choose “export” option as shown below.



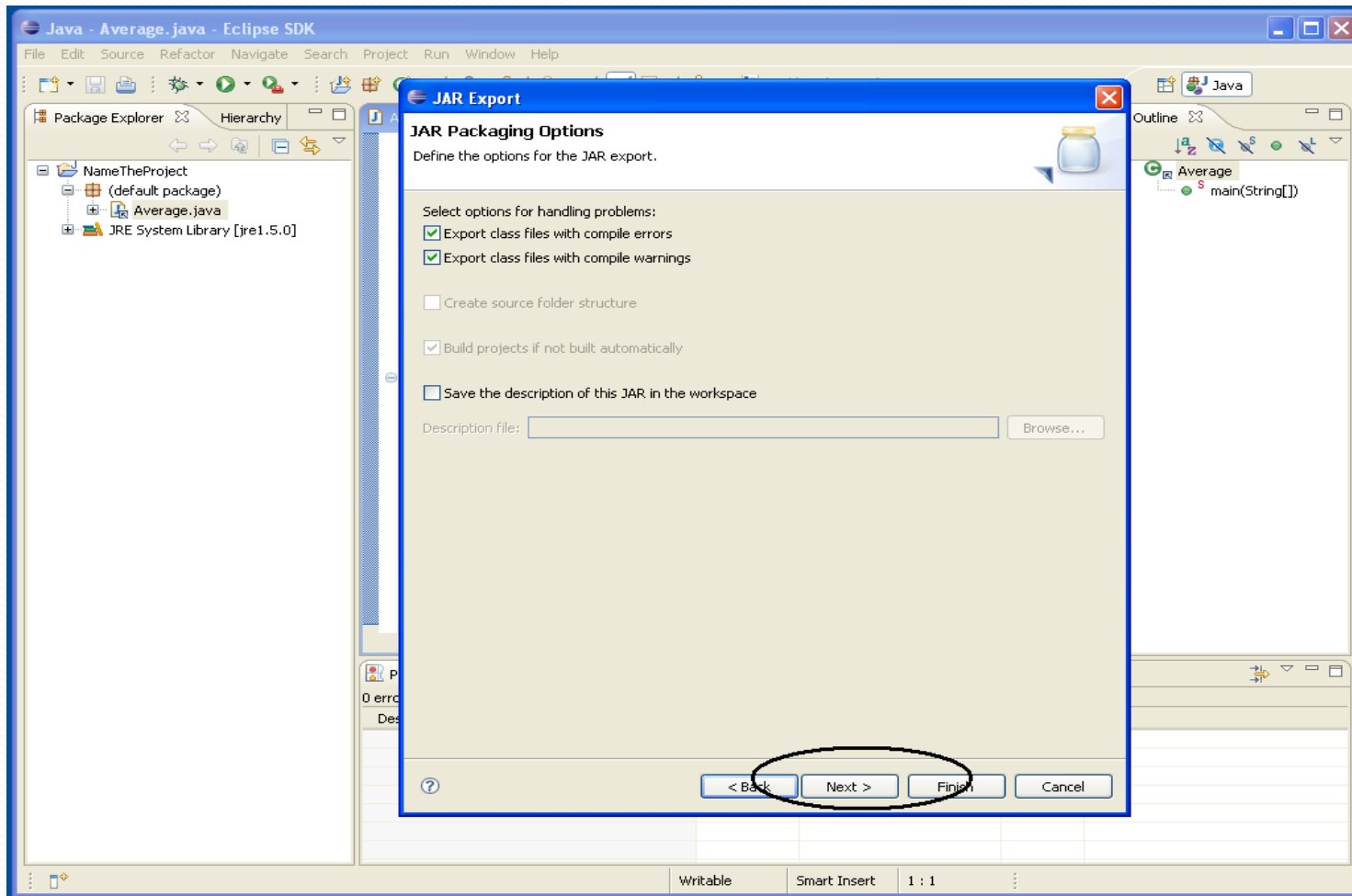
## Step2: Choose “jar file” and “next”.



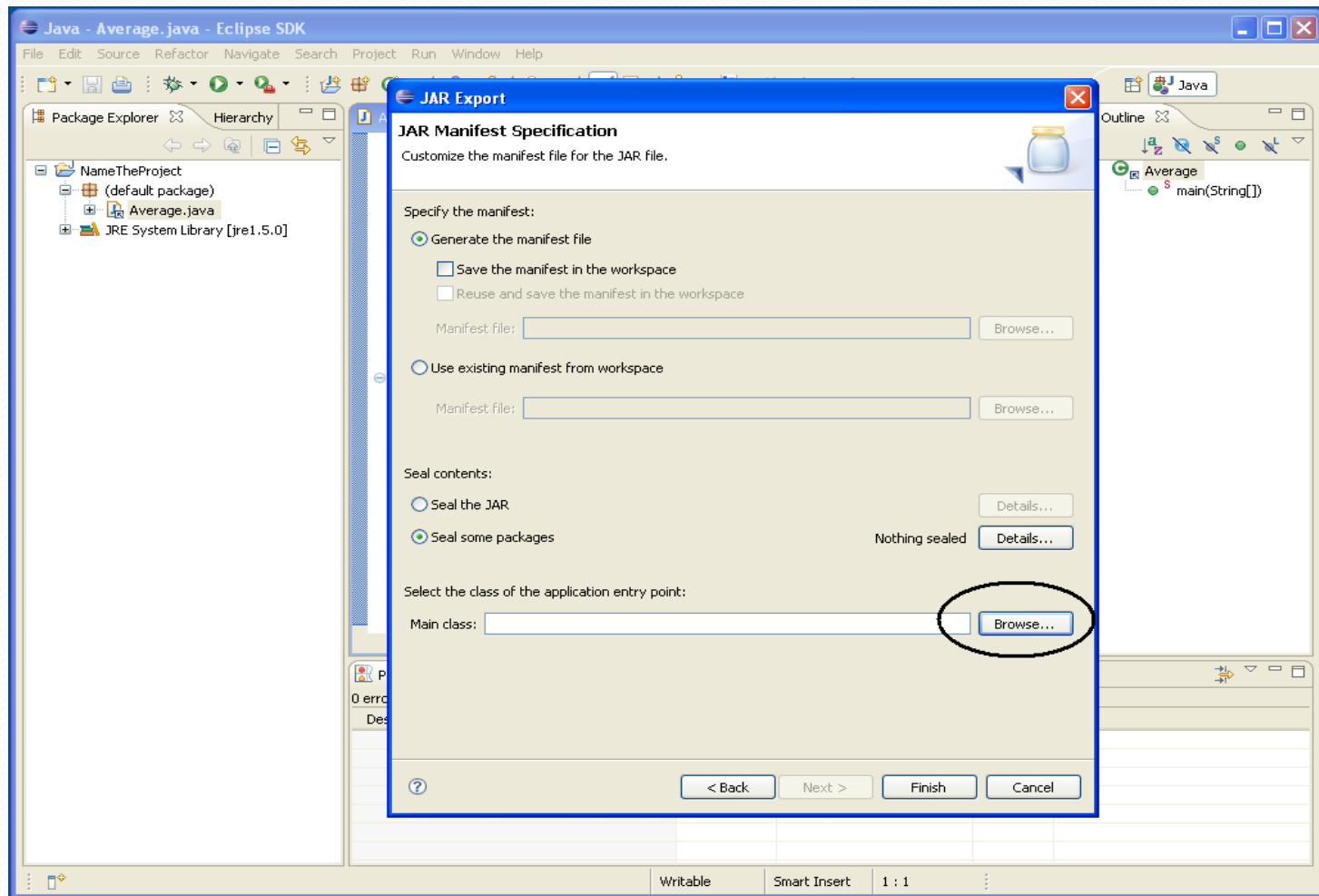
Step3: Select the project and the files you want to add. Browse through and select your destination file.



Step4: In the Packaging options, the first 2 options are usually enough. You can choose the other advanced options depending on your requirements.

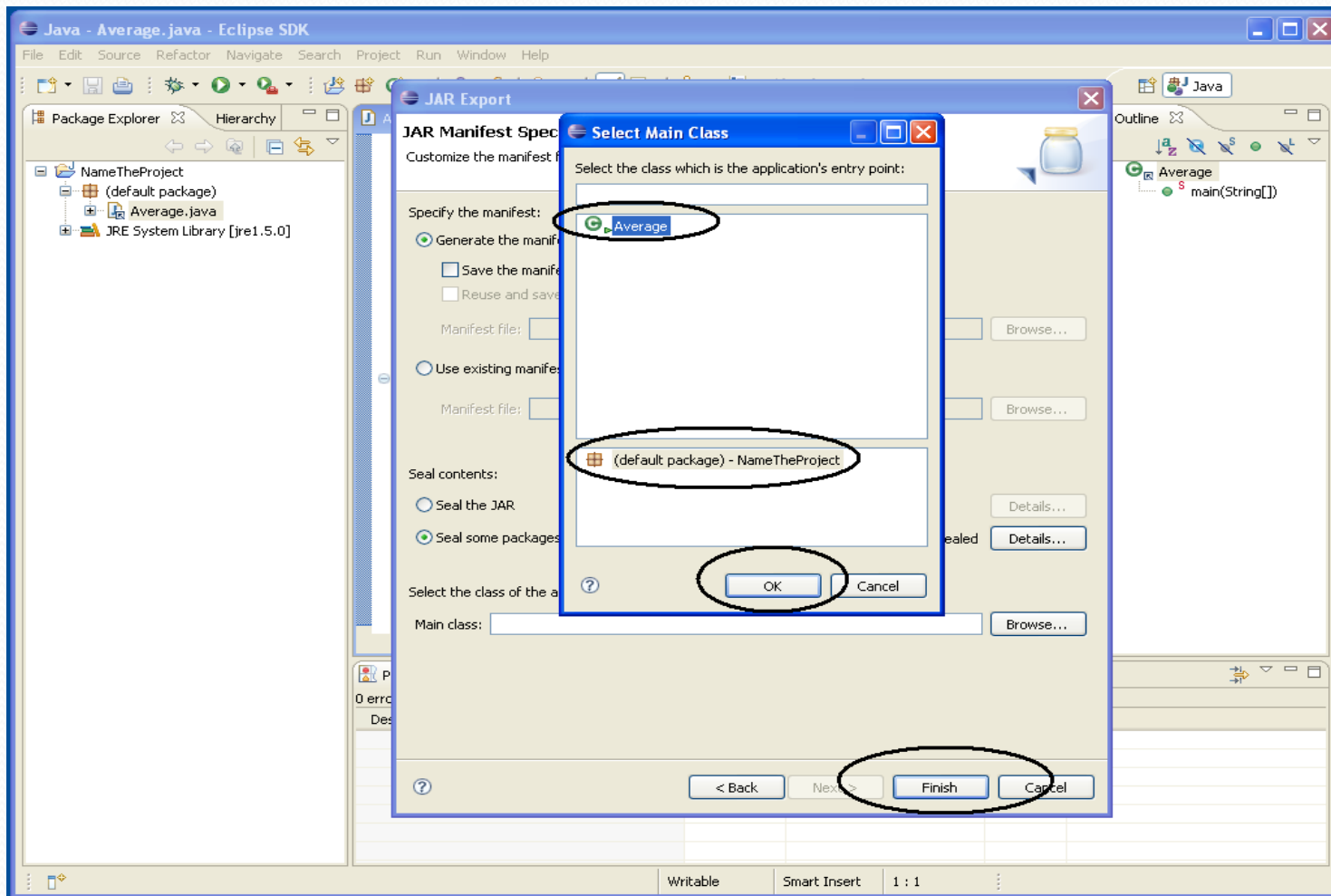


Step5: Jar export. The options chosen here are the standard ones. We then browse through and select the class file which is the entry point. (Read carefully before selecting options).





Step6: 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.



## Step7: Executing a jar file

- The basic command is: `java -jar jar-file`
- If the runtime environment has no information about which class within the jar file is the application's entry point (class containing the main method of your application), you must add a Main-Class header to the JAR file's manifest. See:  
<http://docs.oracle.com/javase/tutorial/deployment/jar/appman.html>

# 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** (<http://help.eclipse.org/mars/index.jsp>).
- Eclipse's GUI builder is a separate component.

# Adding Eclipse plugin

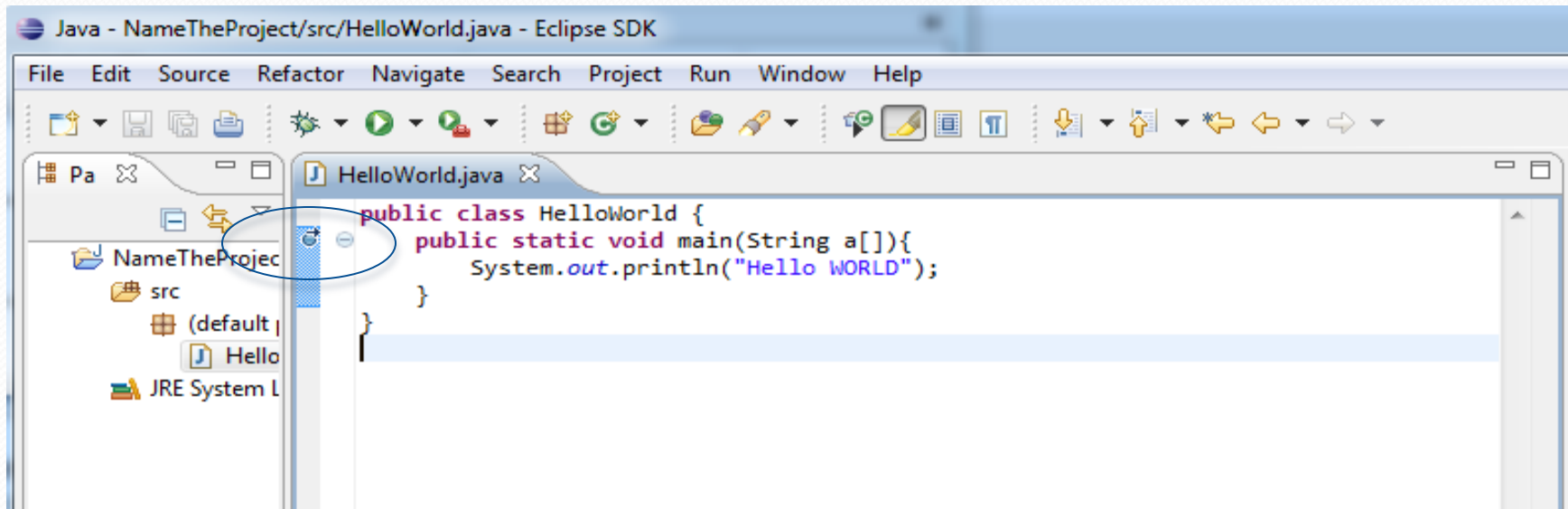
The simplest way is to copy the plugin's folder to the plugins subfolder of the appropriate Eclipse binary folder. This method however, requires that separate copies of plugin binaries be created for different platforms.

# Running 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.



# Debugging

- 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 - click Yes.



# Debug Perspective

The screenshot shows the Eclipse IDE in the Debug perspective. The top toolbar contains several icons for stepping through code, which are circled in blue. A callout box points to the 'Debug' button in the top right corner. The 'Variables' view on the right shows a table with one variable 'a' of type 'String[0]'. The 'Breakpoints' view is empty. The 'Console' view at the bottom shows the output of the program. The 'HelloWorld.java' editor shows the source code with the line `System.out.println("Hello WORLD");` highlighted in blue. The 'Outlin' view on the right shows the class structure.

New Debug perspective – click Java to exit

Buttons to step through the code. Shortcuts available from Run menu

List of breakpoints

Variables in the scope with their current values.

Line of code where we stopped

Output Console

Name	Value
a	String[0] (id=16)

```
public class HelloWorld {  
    public static void main(String a[]){  
        System.out.println("Hello WORLD");  
    }  
}
```

HelloWorld [Java Application] C:\Program Files\Java\jre7\bin\javaw.exe (30/set/2014 13:59:02)

# The famous Dos and Don'ts:

- Set the workspace of 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.