



Lab 1 - An Introduction to Eclipse

Objectives:

- Learn to use Eclipse as Java integrated development environment.
- Practice basic java programming.

Background Knowledge

- Materials from 2140101 - Computer Programming for International Engineer.

Eclipse - an open development platform

Eclipse is an open source community, whose projects are focused on building an open development platform comprised of extensible frameworks, tools and runtimes for building, deploying and managing software across the lifecycle. A large and vibrant ecosystem of major technology vendors, innovative start-ups, universities, research institutions and individuals extend, complement and support the Eclipse platform

In this class, we will use Eclipse as a tool (IDE) for developing JAVA programs.

You will need a Java runtime environment (JRE) to use Eclipse. You may download the Eclipse SDK at <http://www.eclipse.org/downloads/>. You will get a Zip file, uncompress (unzip) it in an empty folder. You are ready to run the Eclipse.

The Workbench

When the Workbench is launched the first thing you see is a dialog that allows you to select where the workspace should be located as shown in Figure 1. The workspace is the directory where your work will be stored.



Your turn ①

Since this is the first time you use Eclipse, you need to create a workspace. Your workspace will have the following name format, `%desktop%\YourID`, where `%desktop%` is the location of your desktop and `YourID` is your student identification. For example, if the desktop of your machine is `"C:\Documents and Settings\Chate\Desktop"`, and your student ID is `"500 12343 21"`, your work space will be

`"C:\Documents and Settings\Chate\Desktop\5001234321"`.

You should create a folder for your work and copy it. We will clean the hard disk everyday. Do not expect your work will be in the computer next time you work in the lab.

You will use the workspace you've just created for all labs in this course.

After the workspace location is chosen, a single Workbench window is displayed as shown in Figure 2. A Workbench window offers one or more perspectives. A perspective contains editors and views, such as the Navigator. Multiple Workbench windows can be opened simultaneously. Initially, in the first Workbench window that is opened, the Java perspective is displayed, with only the Welcome view visible. Click the arrow labeled Workbench in the Welcome view to cause the other views in the perspective to become visible.

(You can get the Welcome view back at any time by selecting **Help > Welcome**.)

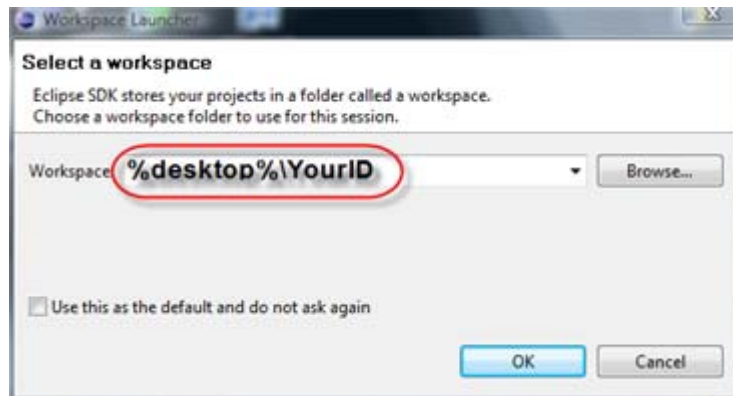


Figure 1 - Eclipse Workspace Launcher



Figure 2 - Eclipse Welcome page

Preparing Eclipse

In this section, you will verify that Eclipse is properly set up for Java development.

Verifying JRE installation and CLASSPATH variables

1. If you still see the Eclipse Welcome page, click the arrow icon to begin using Eclipse.

2. Select the menu item [Window > Preferences...](#) to open the workbench preferences.
3. Select the [Java > Installed JREs](#) preference page to display the installed Java Runtime Environments as shown in Figure 3. Confirm that a JRE has been detected. By default, the JRE used to run the workbench will be used to build and run Java programs. It should appear with a checkmark in the list of installed JREs. We recommend that you use a Java SDK instead of a JRE. An SDK is designed for development and contains the source code for the Java library, easing debugging. Additional SDKs can be added by searching the hard drive for installed SDKs. To do so, simply click the **Search...** button and specify a root folder for the search. **Select the most recently version of the JRE.**

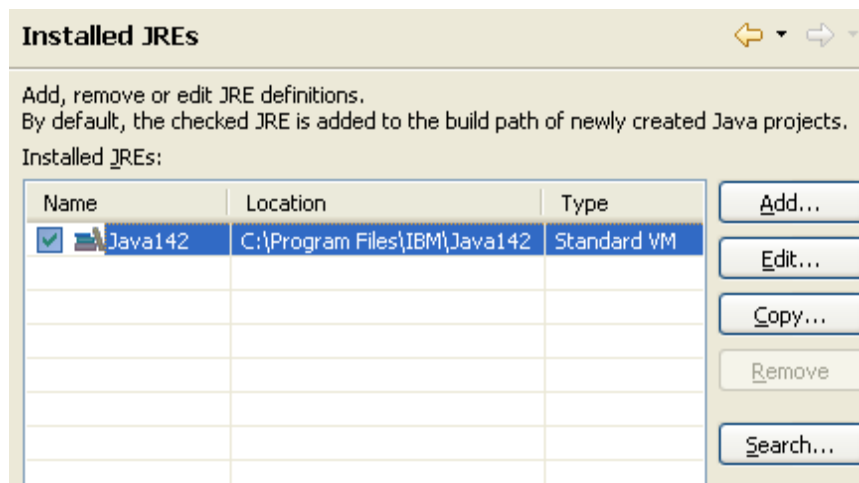


Figure 3 - Installed JREs preference page

4. Select the [General > Workspace](#) preference page as shown in Figure 4. Confirm that the **Build automatically** option is checked.

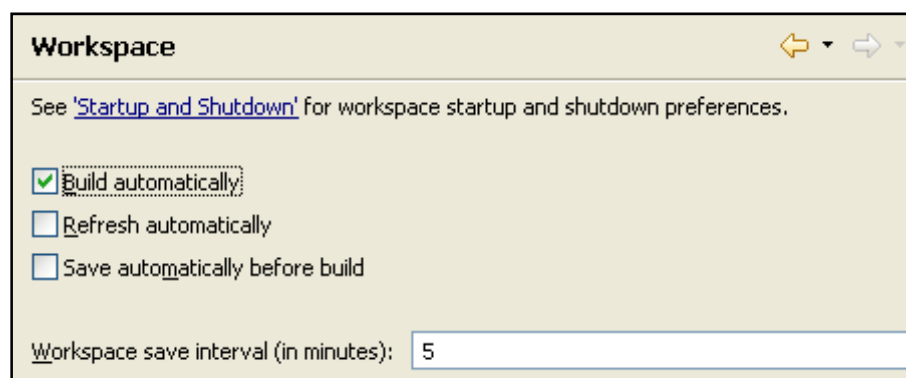


Figure 4 - Workspace preference page

5. Select the [Java > Build Path](#) preference page as shown in Figure 5. Confirm that **Source and output folder** is set to **Project**.

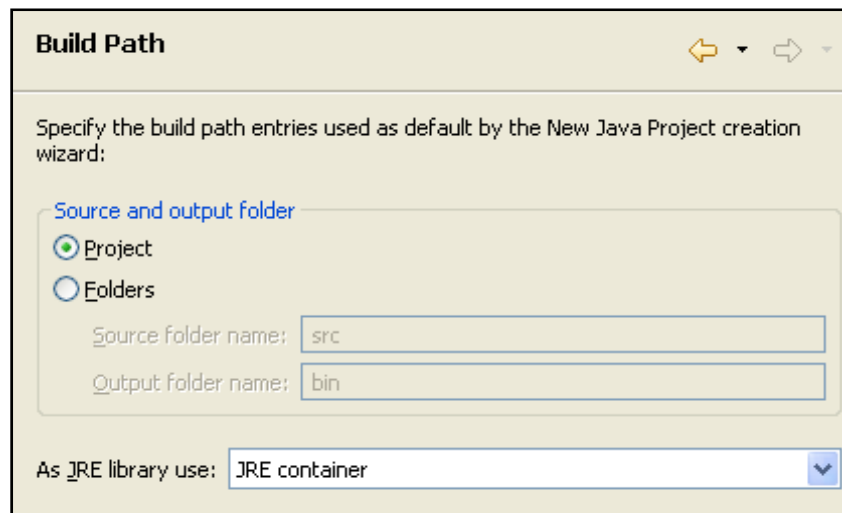


Figure 5 - Java's Build Path preference page

6. Select the [Java > Editor](#) preference page as shown in Figure 6. Confirm that option **Report problems as you type** is checked.

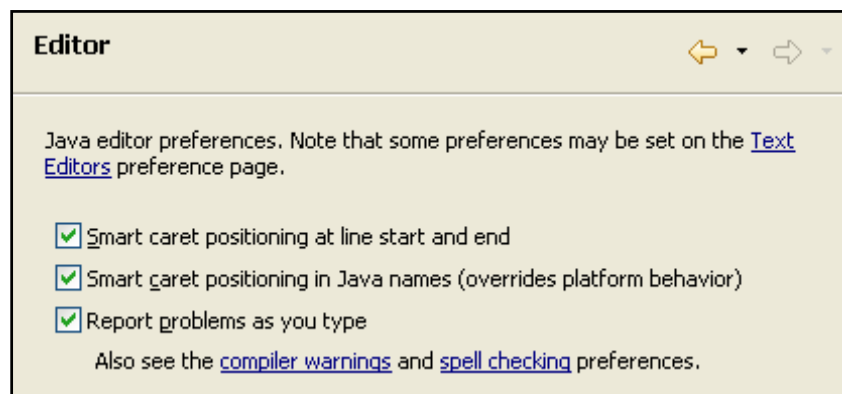


Figure 6 - Java's Editor preference page

7. Click on **OK** to save the preferences.

Creating your first Java project



Your turn ①

Complete the following steps.

1. *Creating a project.*

Before we start working on a program, we must create a project for it. An Eclipse project helps us organize our files, so that we do not mistakenly work on the wrong program.

- 1.1. To create a project, go to **File** and choose **new -> project** as shown in Figure 7.
- 1.2. Since we are writing a Java program, choose **Java Project**.
- 1.3. Then type in your project names. Let's name it **Lab1v1**. The name that we use is case sensitive.

1.4. At this point, we have two choices, **next** and **finish**. This is a simple project, so let's choose **finish**. Our project will then be displayed on the left hand window. The window is divided into 4 parts as shown in Figure 8. We can extend or move all parts to anywhere we want.

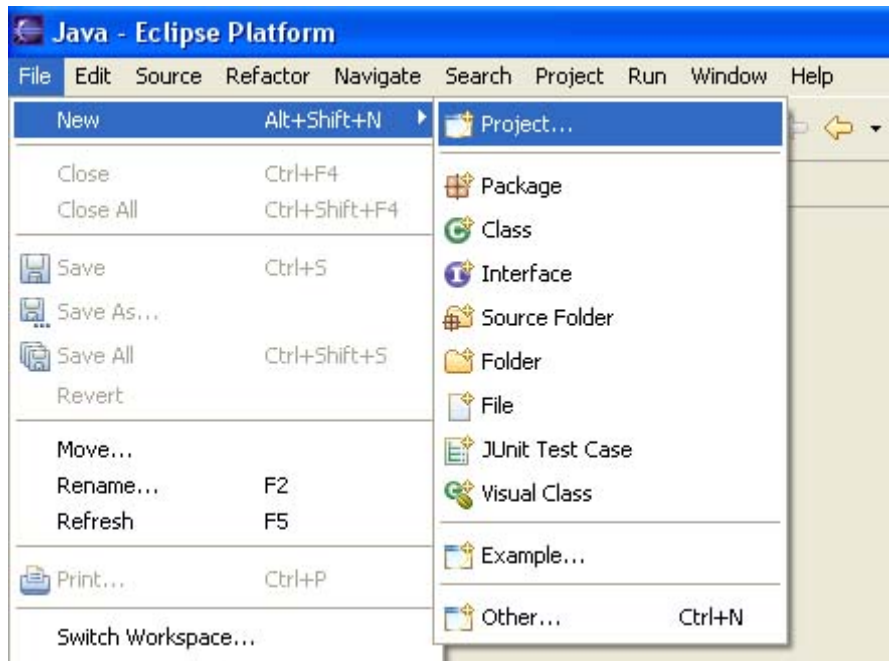


Figure 7 - Create a new project

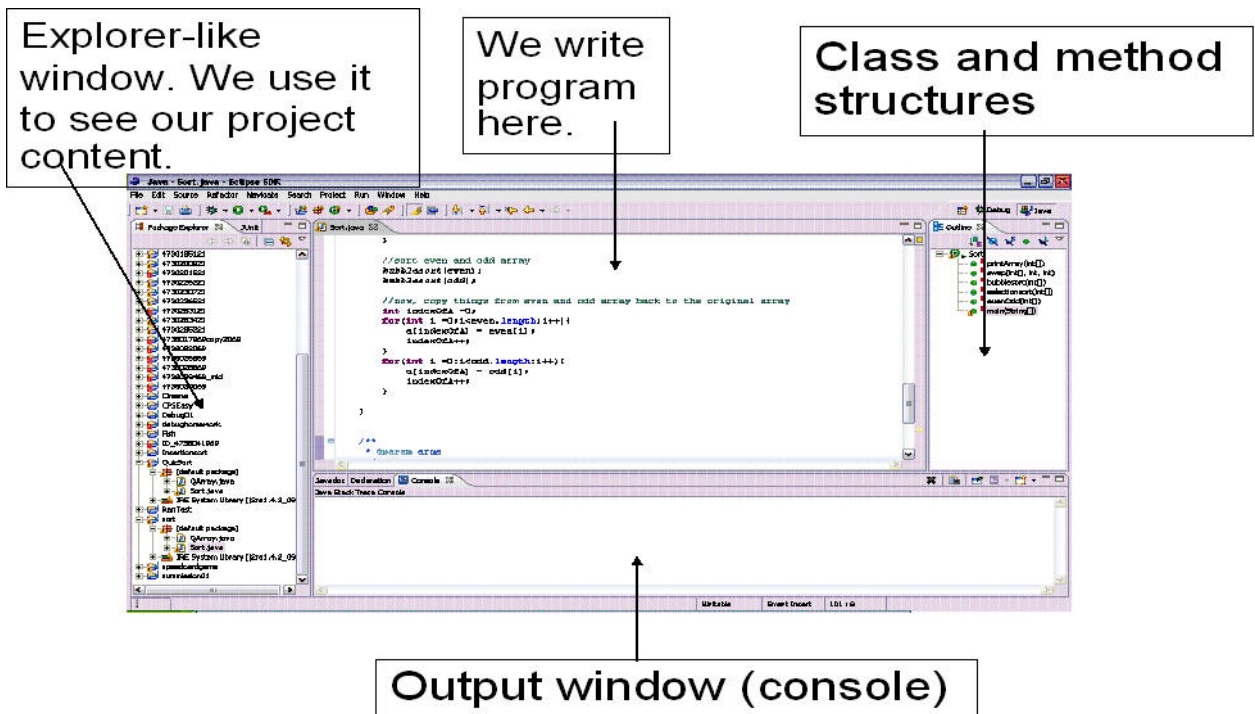


Figure 8 - Eclipse IDE

Tip: Right clicking in the Eclipse Explorer window will also get you a pop-up menu for creating your project.

9. *Creating your file.*

- 9.1. Try creating your class (yes, it's your program) by choosing **File-> New ->Class**. You can also do this by selecting from pop-up menu if you right click in the explorer window.
- 9.2. A new Java class dialog will pop up, prompting you to fill in your class information as shown in Figure 9.
- 9.3. Let's create a class called "**Hello**". Let this class have its main method. Do not select **Inherit abstract method**. You can simply select the check boxes. Eclipse will generate your main method structure for you, with lots of comments. You can leave those comments in.
- 9.4. Let's write a program to print the word "Hello!" This is easy, but you've got to get used to Eclipse, **so do it now**.
- 9.5. While you are writing your program, you may notice that your file name has a * in it. This means your file has not been saved.

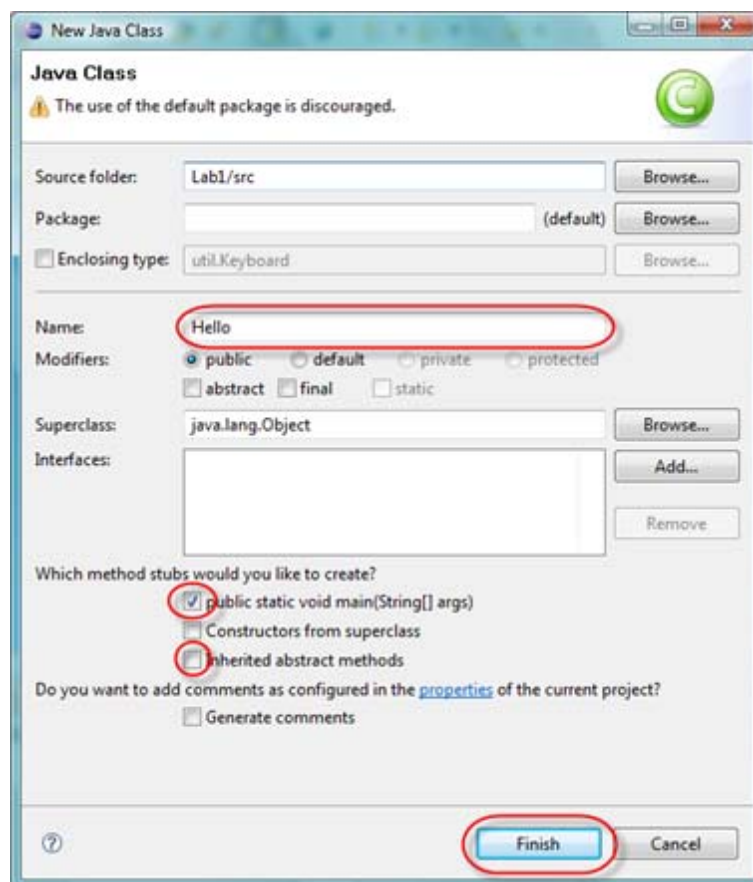


Figure 9 - New Java Class dialog

- 9.6. When you finish writing your program, try to run it without saving (select **Run -> Run As Java Application**). Eclipse will prompt you to save your file at this point. What you print will be shown in Eclipse's console window.

NOTE: Sometimes if your program won't run, try to save it manually first. This may fix the problem.

10. Importing file.

Sometimes you won't be writing your program from scratch. Importing is a way to copy files into your project. You can do this by right click on your project name, and then choose import. Eclipse will pop a form prompting you to choose what kind of information will be imported as shown in Figure 10. We will mainly be using 2 import sources:

- file system - normal copying of files into your project.
- archive file - import an unzip the *.zip* or *.jar* file (jar is Java's zip format) for you.

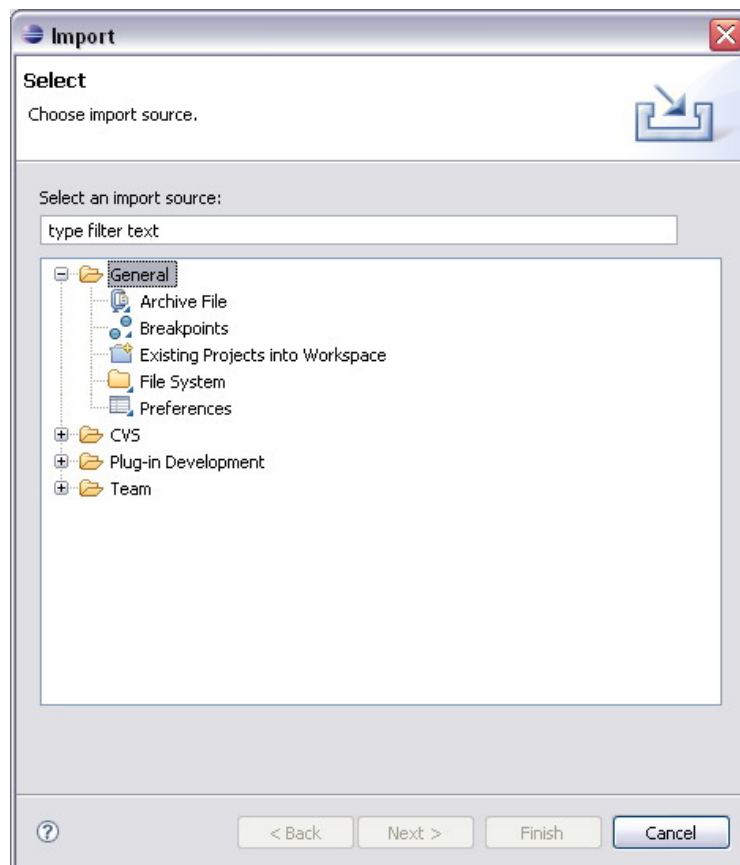


Figure 10 - Import dialog

- 10.1. Try importing the `InsertionSort.jar` file. (You may ask Lab. Instructor for a copy of `InsertionSort.jar`)
- 10.2. Now, unzip the jar to any other folder, create a new project called **Lab1v2** and try to import the unzipped file.

11. Creating .jar file

Actually, jar (java archive) is zip, but we can also make .jar file executable. This will be useful when you want to deploy your program in real use.

- 11.1. To create a jar file, right click on your project, and then select **Export** as shown in Figure 11. Choose **JAR** file, name it (click **Browse** and change the folder to Desktop, and type in ***yourname.jar***), and select all options as shown in Figure 12 then click **Next**.
- 11.2. JAR Package option dialog is shown, click **Next**.
- 11.3. JAR Manifest Specification dialog is shown as in Figure 13. Click **Browse**.

- 11.4. Select main class window is shown. Select **InsertionSort** and then click **OK** as shown in Figure 14.
- 11.5. Click **Finish**.
- 11.6. Go to Windows desktop and verify that your JAR file created in step 11.1 exists. Double-click that file to run the executable jar file.

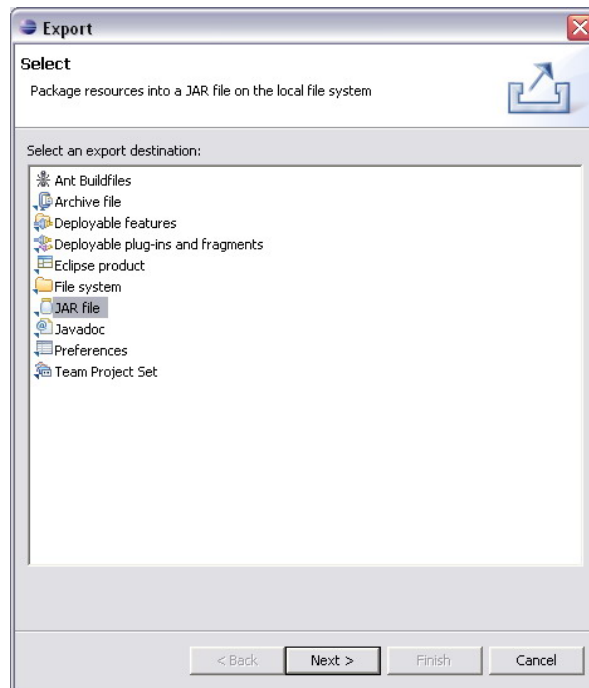


Figure 11 - Export dialog

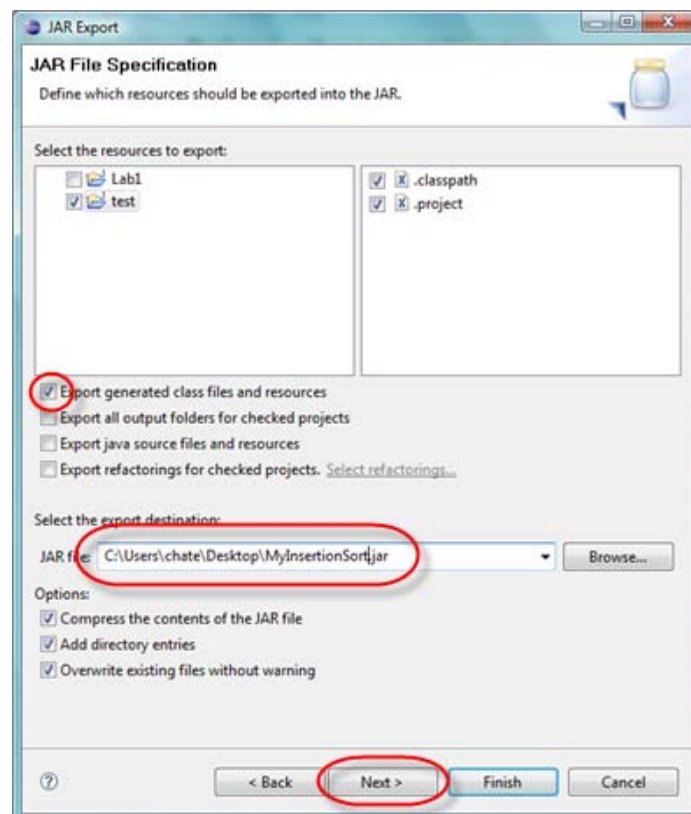


Figure 12 - Export JAR file

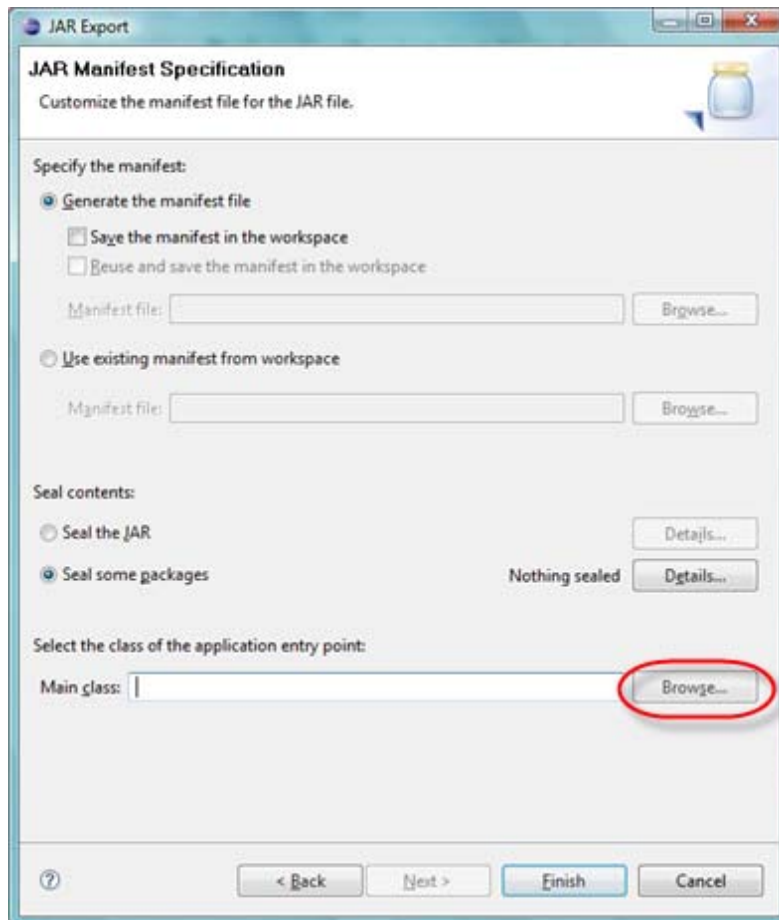


Figure 13 - JAR Manifest Specification

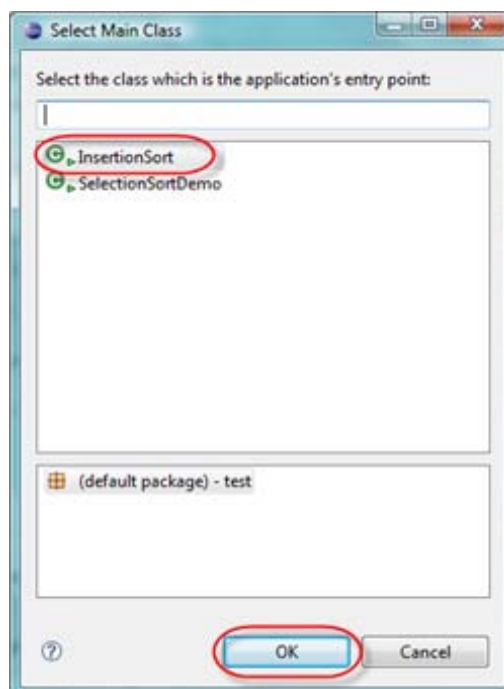


Figure 14 - Select main class window

NOTE:

- When you submit your homework, use .jar format, you must choose to export generated class files and resources and export Java source files and resources.
- Before sending your jar file to anywhere, try unzipping it to test if the jar creation is correct.

Tips & Tricks

1. To let the editor show line numbers:
 - 1.1. Go to menu **Window**, choose **Preferences**.
 - 1.2. At **General -> Editors -> Text Editors**, check show line numbers as shown in Figure 15.

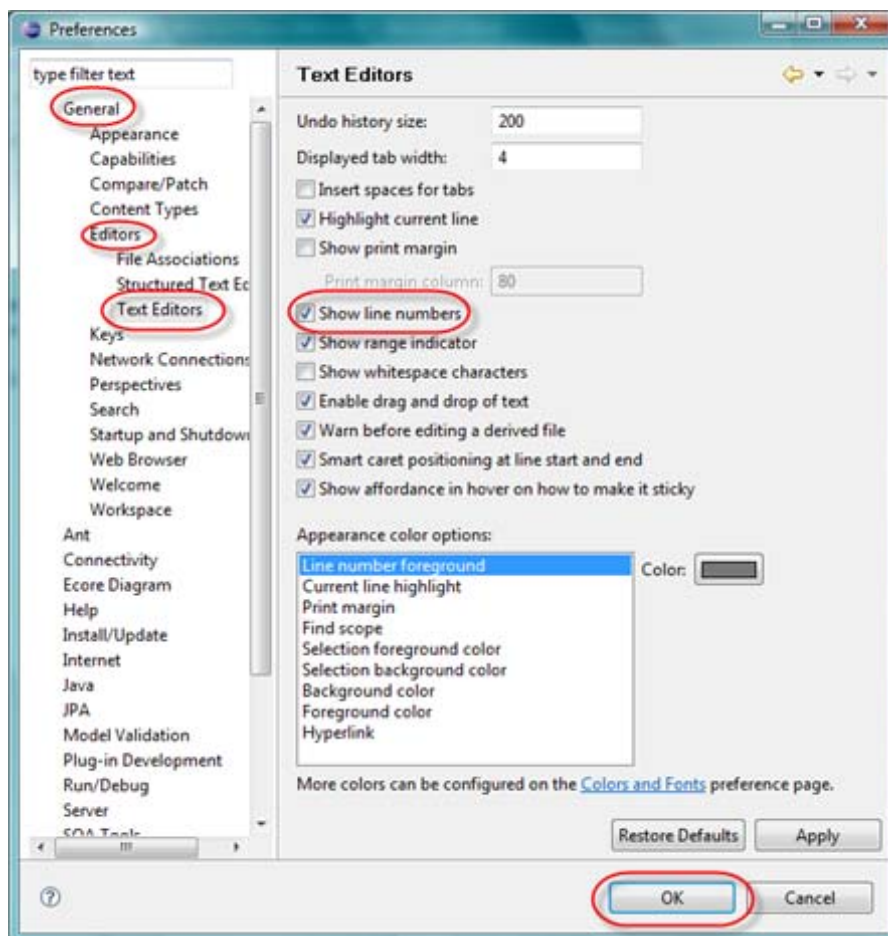


Figure 15 - Text Editors preference page

2. If you want to find a bracket's corresponding pair, move the cursor behind the bracket. Its pair will be shown.
3. Use **Ctrl+Shift+F** to layout your code beautifully.
4. If you are typing a long name, while typing, press **Ctrl+Space**. Eclipse will fill the rest of the name for you or present choices for you to choose.
5. At Window menu, choose Show View. There are different views for looking at your project. We will introduce 2 of them here:
 - 5.1. package explorer - shows only parts related to writing programs.
 - 5.2. navigator - shows everything in the project directory.

6. If you want to change a variable name or file name, use **Refactor**. It is better than search and replace because it updates all relationships with other project components for you. To use refactor, select project name, class name, or anything that you want to rename, and right-click then select **Refactor -> Rename** as shown in Figure 16.

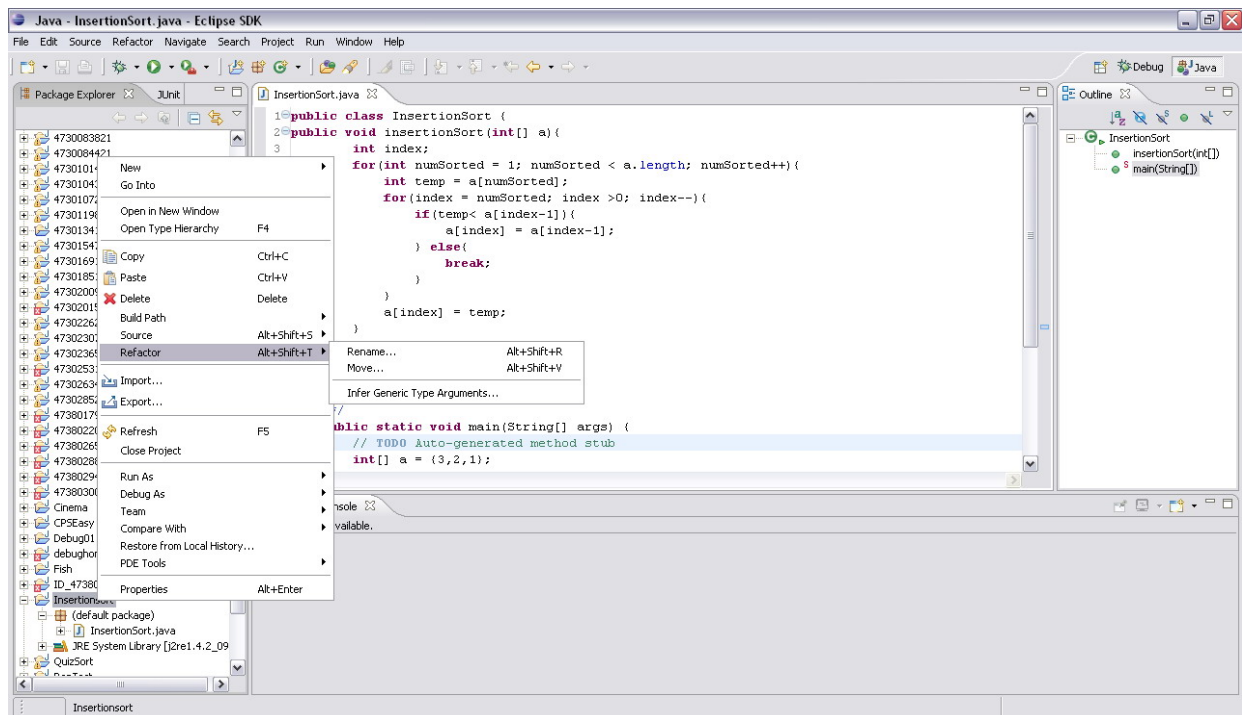



Figure 16 - Use Refactor to rename everything

7. When you are trying to run a program, make sure you are focusing on its file.
8. To run the program you run recently, click on .
9. When you are deleting a project, you can choose to delete content (delete all the files in project folder). If you do not choose to delete content, Eclipse simply makes the project invisible (the real files still exist).

Java review

Console Keyboard Input

```
import java.io.*;

public static void main(String[] args) throws IOException {
    // define buffered for keyboard input
    BufferedReader kbd = new BufferedReader(
        new InputStreamReader(System.in));

    // use readLine() to read a string from keyboard
    System.out.print("Enter your input: ");
    String text = kbd.readLine();
    // convert input string into an integer
    int intInput = Integer.parseInt(text);
}
```

Keyboard Input Dialog

showInputDialog

```
public static String showInputDialog(Object message,  
                                   Object initialValue)
```

Shows a question-message dialog requesting input from the user, with the input value initialized to `initialSelectionValue`. The dialog uses the default frame, which usually means it is centered on the screen.

Parameters:

`message` - the Object to display

`initialSelectionValue` - the value used to initialize the input field

```
import javax.swing.JOptionPane;  
public static void main(String[] args) {  
    // use showInputDialog to get input  
    String text = JOptionPane.showInputDialog("Enter your input: ");  
}
```

Lab 1 Exercise



Your turn

NOTE:

- *This is the first lab. Therefore it will be MUCH easier than future labs. Do not take it for granted.*
- *Prepare yourself by review material from previous semester.*
- *Be on time.*
- *You can help each other, but don't copy.*
- *Do not hesitate to ask teacher and TA. (Try to solve the problem by yourself first.)*
- *Use Java's help from java.sun.com or j2se6.chm.*
- *When you finish each exercise, you will ask a lab's instructor to verify the results and sign your lab. results' form. You should create a new project and new folder for each exercise.*

1. Write a Java program that prints to the screen all prime numbers from 2 to 1000, ten numbers per line.
2. Write a Java program that prints out all elements of a one-dimension array of `int`. For example, if an array of `int` has 3, 5, 2, 6, 3 as its elements, the program should print { 3, 5, 2, 6, 3 }
3. Write a Java program that read 2 strings. The program will print "True" to the screen if all characters in the first string are in the second string (regardless of case), print "False" otherwise.

For example, if the first string is "**This is it.**" and the second string is "**It is the answer**", the program will print "**True**". However, if the first string is "**It is the answer**" and the second string is "**This is it**", the program will print "**False**", since there is no '**a**' or '**w**' or '**r**' in the second string.



Lab 1 – Introduction to Eclipse.

Task	Description	Result	Note
1	Create workspace		
2	Create a new Java Project		
3	Import <ul style="list-style-type: none">• .jar (or .zip) file• Unzipped file(s)		
4	Print prime numbers		
5	Print array elements		
6	Compare two Strings		
7			