

Lab Exercise: IDE Familiarization





About Intertech

Thank you for choosing Intertech for your training. The next page of this document will get you started with your lab if you'd like to skip ahead. Below is a brief overview of Intertech as well as a promo code for you for future live Intertech trainings.

Intertech (www.intertech.com) is the largest combined software developer **training** and **consulting** firm in Minnesota. Our unique blend of training, consulting, and mentoring has empowered technology teams in small businesses, Fortune 500 corporations and government agencies since 1991.

Our training organization offers live in-classroom and online deliveries, private on-site deliveries, and on-demand options. We cover a broad spectrum of .NET, Java, Agile/Scrum, Web Development, and Mobile technologies. See more information on our training and search for courses by [clicking here](#).

We appreciate you choosing Intertech!

Lab Exercise

IDE Familiarization

By now you are probably more than eager to get your first Java program rolling. The purpose of this first simple lab is to get you used to your surroundings. The first program you will write is, of course, HelloWorld. Although the code is simple, there are many things that can go wrong when compiling and running a Java application, especially on your first try. After getting your first Java application running, you will move it to an industrial-strength integrated development environment (IDE).

Specifically, in this lab you will:

- Write a simple Java program
- Compile your Java code from the command line using the JDK tool `javac.exe`
- Run your program using the JVM `java.exe`
- Start a Java IDE called Eclipse
- Copy your code to Eclipse
- Compile and execute your code in Eclipse

Writing Java Using Only a Text Editor

For most of this class, you will be encouraged to use a powerful editor called Eclipse. For the first lab, you will be asked to use a simpler editing tool (any text editor of your choice – as long as it's able to save the code as “plain text.”) and the JDK tools `javac.exe` and `java.exe`. While an IDE makes it much easier to write, compile, debug, and execute code, having an understanding of the engines that work under the hood of an IDE is important.

Step 1: *Create HelloWorld.java*

1.1 Open your text editor.



If you aren't able to save the file as a “plain text” file with your favorite editor, use one of the following:

Mac: Use “TextEdit.” Before writing your code, select Format -> Make Plain Text. Then select Edit -> Substitutions... and un-check “Smart Quotes.”

Windows: Use “Notepad” (not “Wordpad”), and when saving the file select “Save as type: All Files (*.*)” and Encoding: ANSI.

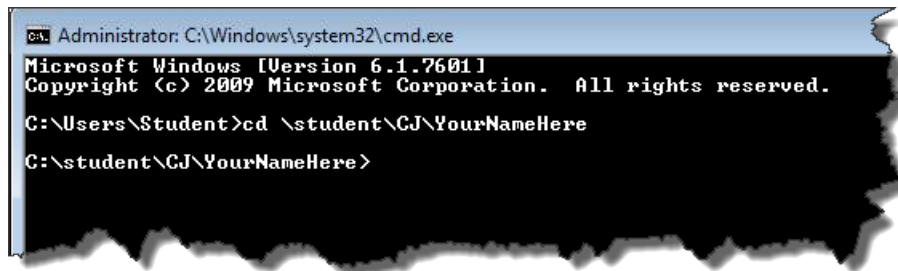
1.2 In the editing window, add the following code.

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

- 1.3 Save this file in a file called HelloWorld.java.

Step 2: Compile HelloWorld

- 2.1 If using windows, open a command prompt. Otherwise, if you are using a Mac, open up the terminal.
- 2.2 Using the “cd” command, navigate to the directory that you saved the file to.

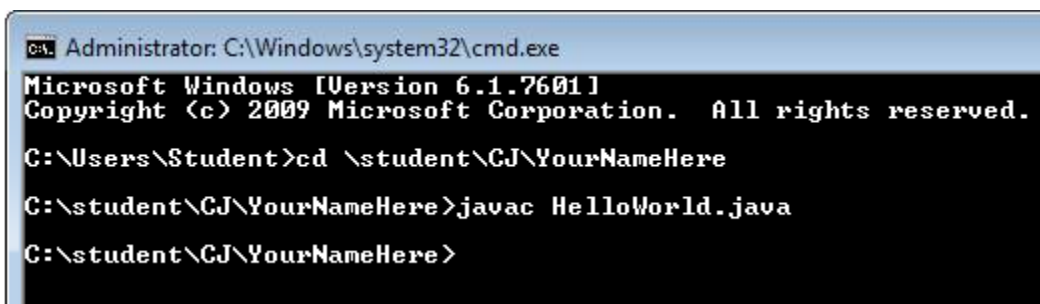


```
C:\> Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Student>cd \student\CJ\YourNameHere
C:\student\CJ\YourNameHere>
```

- 2.3 Compile your application. Run **javac** to compile your program. To run javac, type in “javac” followed by the name of the file (HelloWorld.java) you want to compile.

```
javac HelloWorld.java
```

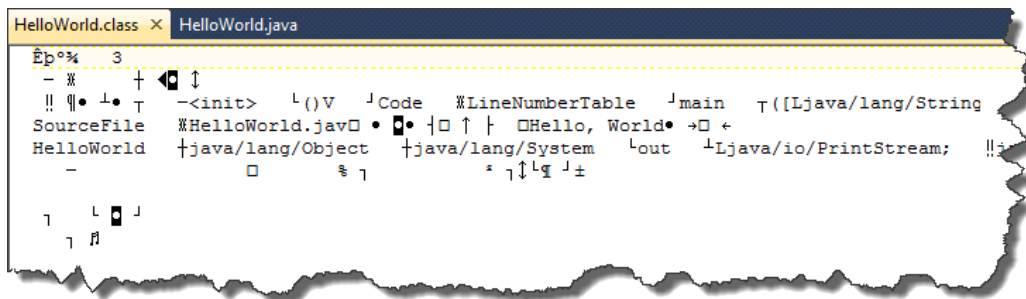
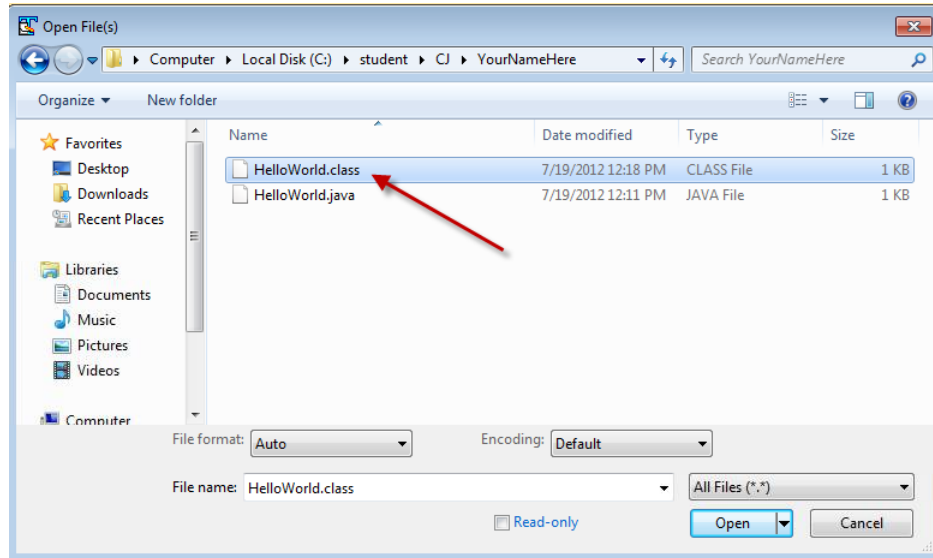


```
C:\> Administrator: C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Student>cd \student\CJ\YourNameHere
C:\student\CJ\YourNameHere>javac HelloWorld.java
C:\student\CJ\YourNameHere>
```

- 2.3.1 If you have compiler errors, you will need to fix them in your text editor and recompile.

- 2.4** Check out the bytecode file. You should have a file called HelloWorld.class stored in the same folder as your HelloWorld.java file. This is the “bytecode” or “class file.” This file is created by the compiler. Don’t edit it, but you can open it with a text editor.

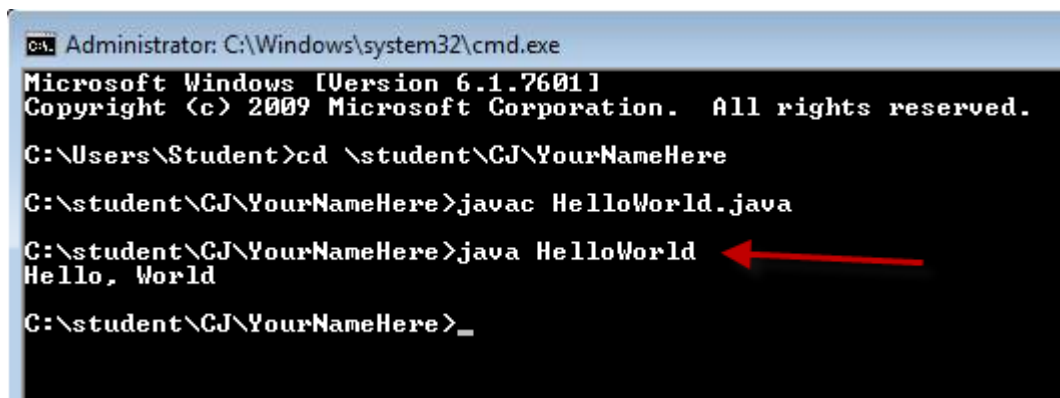


Step 3: *Run HelloWorld*

- 3.1 From the command prompt, type in “java” followed by the name of the class.

```
java HelloWorld
```

- 3.2 This should run the program, producing the following result:

A screenshot of a Windows command prompt window. The title bar reads "Administrator: C:\Windows\system32\cmd.exe". The window content shows the following text:

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Student>cd \student\CJ\YourNameHere
C:\student\CJ\YourNameHere>javac HelloWorld.java
C:\student\CJ\YourNameHere>java HelloWorld
Hello, World
C:\student\CJ\YourNameHere>_
```

A red arrow points to the output "Hello, World" on the line following the "java HelloWorld" command.

Step 4: *Rejoice*

Stand up and fire one arm in the air while screaming “I did it!” Well, you actually don’t need to be quite so obvious, but at least give yourself a pat on the back. You are now officially able to program in Java. Learning the basics of a new environment is important and significant. From now on, if you are asked to compile a Java application, you can. If you are asked to run a Java application, you can.

Using Eclipse

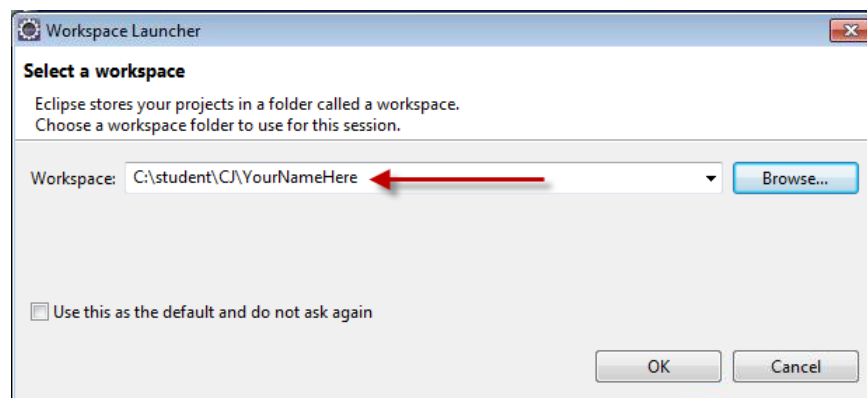
The use of the command prompt is a great exercise, but most developers don't see it on a day-to-day basis. Most Java developers are more likely to code using an IDE, the most popular of which is Eclipse (<http://www.eclipse.org>). This open source (free) Java editor provides powerful features and helps you write good Java code. In this lab, you will be introduced to Eclipse, which you can and should use for all future labs.

Step 5: Begin working with Eclipse

5.1 Launch Eclipse (if you haven't downloaded and installed this yet, review the instructions in the first section of this course).

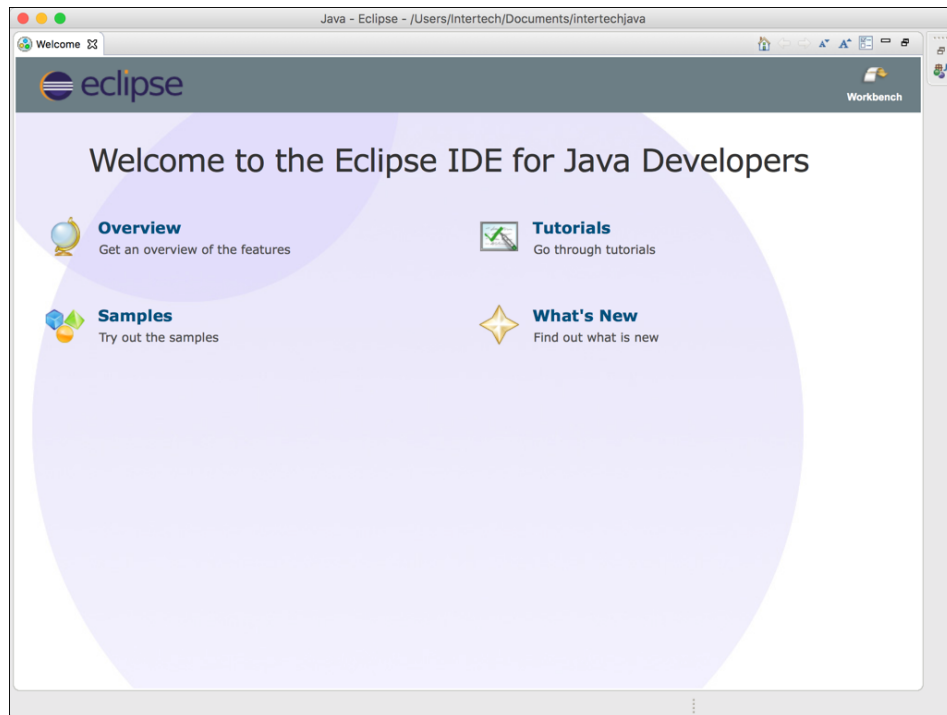
5.1.1 Next, you will see the splash screen for a few seconds.

5.1.2 You will be asked what workspace to use. Choose any name and location that you'll easily find in the future.

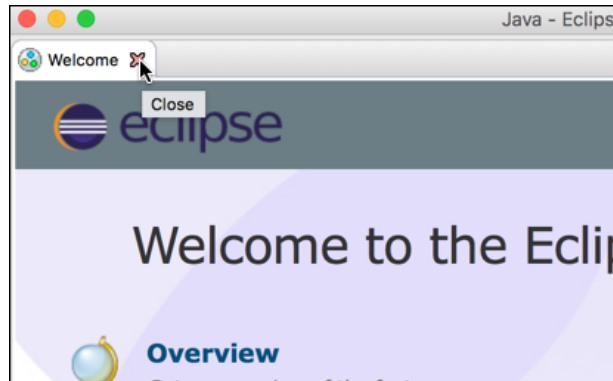


Don't check the "Use this as the default and don't ask again" check box. Changing workspaces is fairly common and might happen in this class. After clicking the *OK* button, the splash screen will provide a status update.

In a few seconds, Eclipse opens.



5.1.3 Close the Welcome tab to see the actual Eclipse IDE by clicking on the 'X' on the Welcome Tab.



5.2 Make a "Java Project" to store your classes and code.

5.2.1 Select *File > New > Java Project ...*

5.2.2 In the ensuing window, enter HelloWorld as the project name and click the *Finish* button.

New Java Project

Create a Java Project
Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location

Location: [Browse...](#)

JRE

☒ Use an execution environment JRE: [Configure JREs...](#)

☐ Use a project specific JRE:

☐ Use default JRE (currently 'Java SE 8 [1.8.0_91]')

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

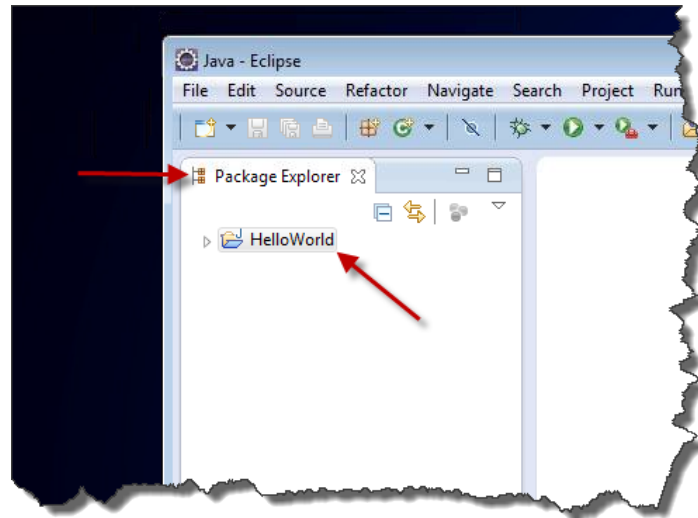
Working sets

☐ Add project to working sets

Working sets: [Select...](#)

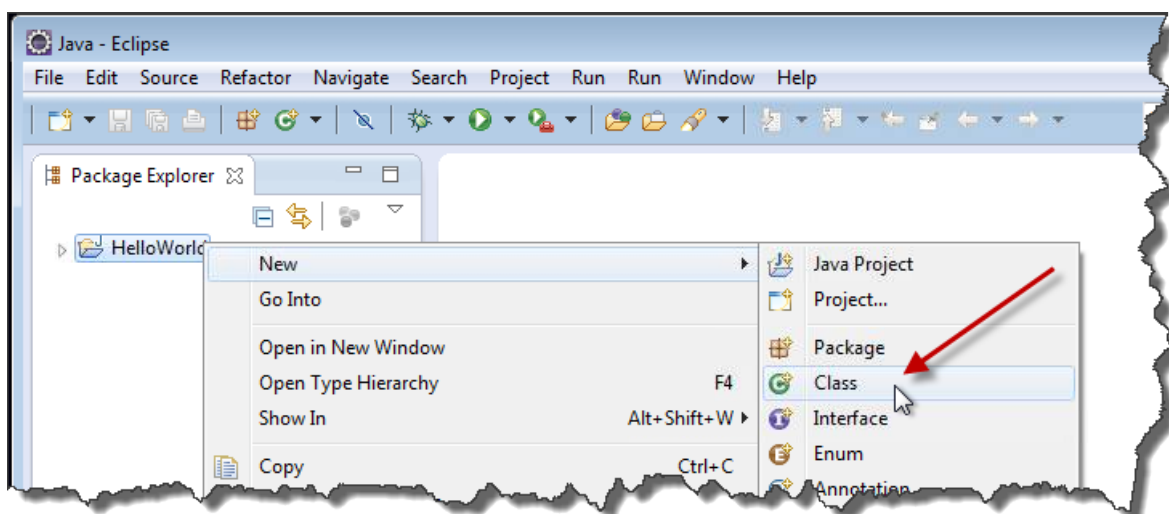
[? < Back](#) [Next >](#) [Cancel](#) [Finish](#)

This should result in a new project (HelloWorld) being displayed in the Package Explorer view (frame).

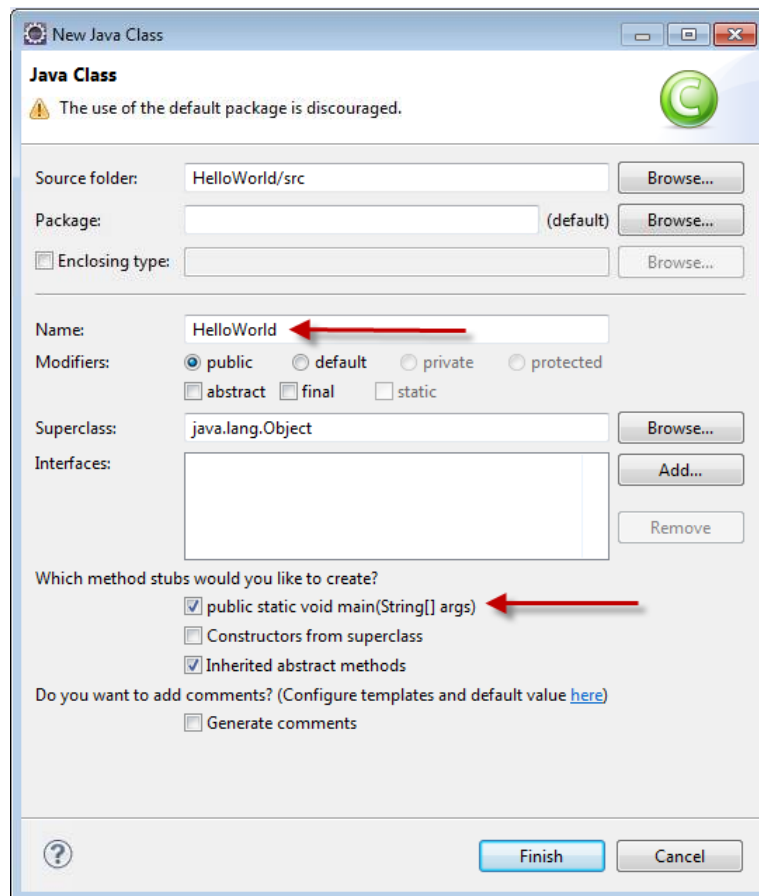


5.3 Make a new Java class.

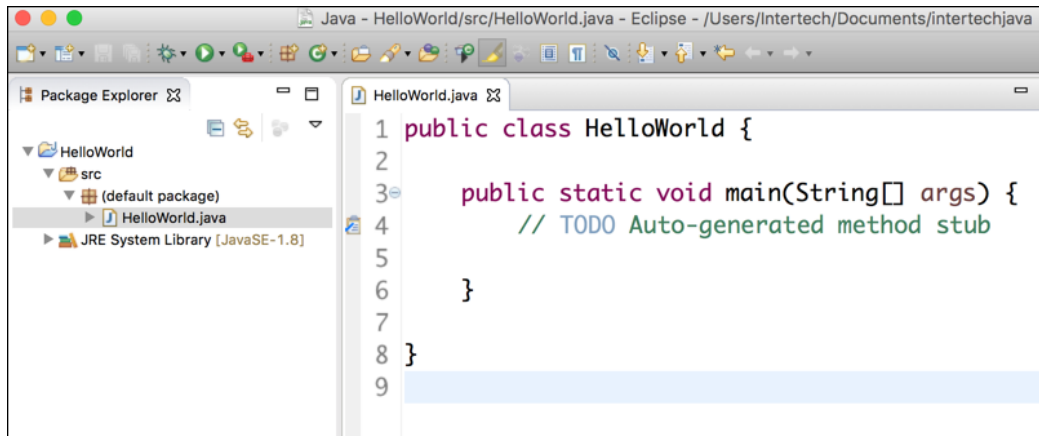
5.3.1 To make a new Java class, simply use the file menu or right-click on the HelloWorld project and select *New > Class*.



5.3.2 In the dialog window that displays, enter HelloWorld as the name of the class, and check the box to have the IDE create a main method. Then click the *Finish* button.



This should create a HelloWorld.java file in the src folder of the project (under the default package) and open a HelloWorld.java editor for you to add code.

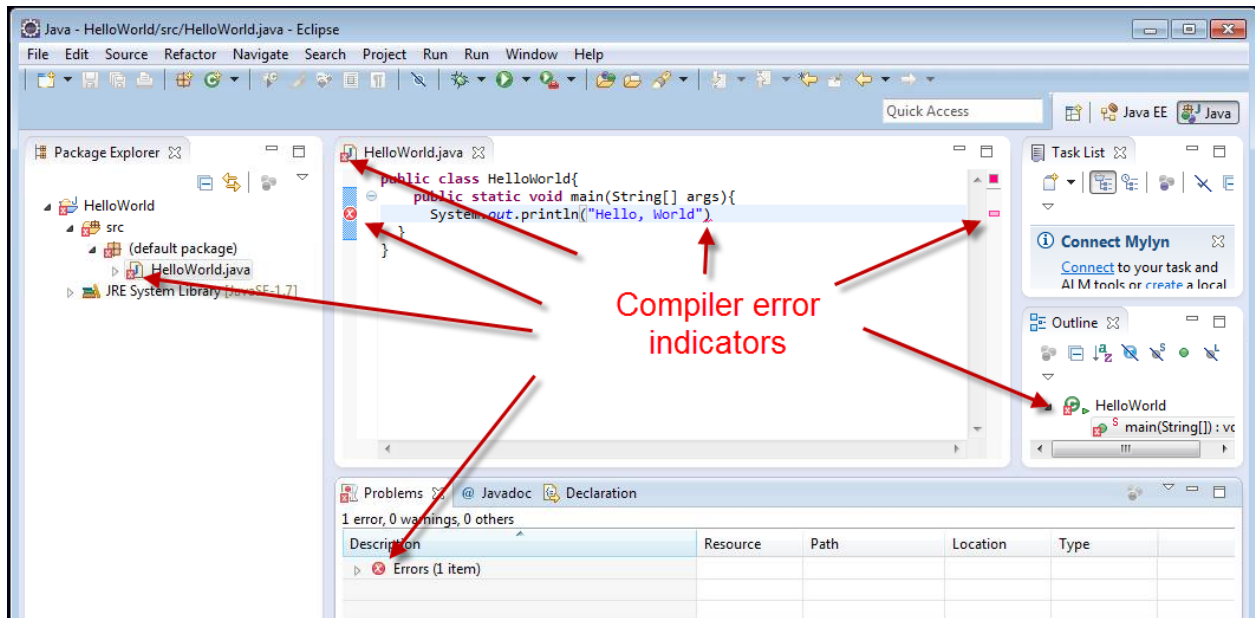


5.3.3 Reenter the HelloWorld code as you did in your text editor, and save the file by selecting *File > Save* (or Ctrl-S) from the IDE's menu bar.

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

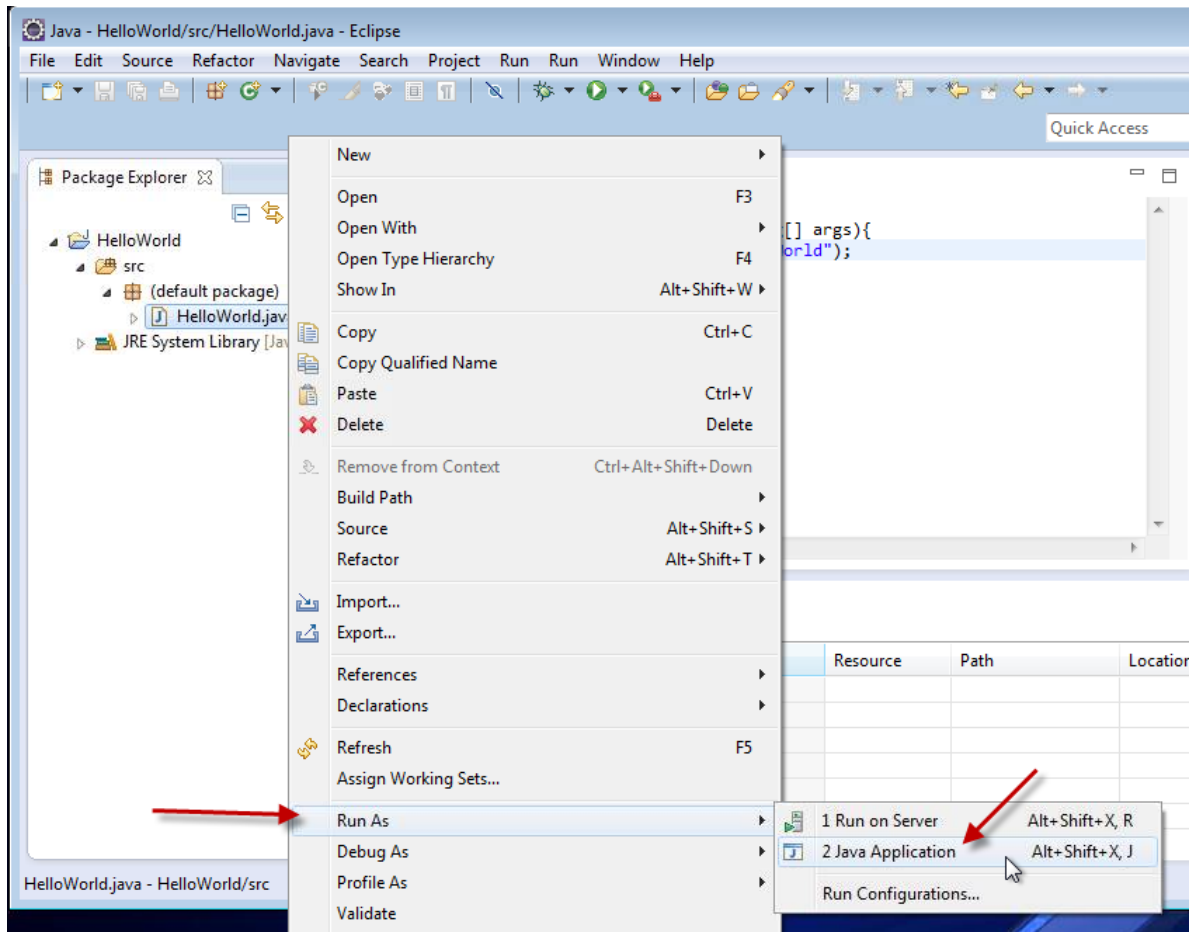


Note: When you save a Java class in Eclipse, Eclipse automatically compiles it. If your code generates a compiler error, it shows up in many places (as shown in the picture below).

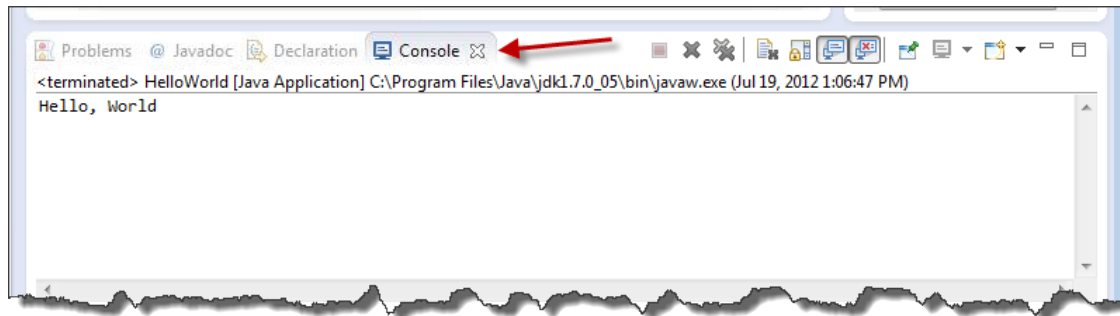


5.4 Edit and fix any compiler errors. Resave your file until you have no errors.

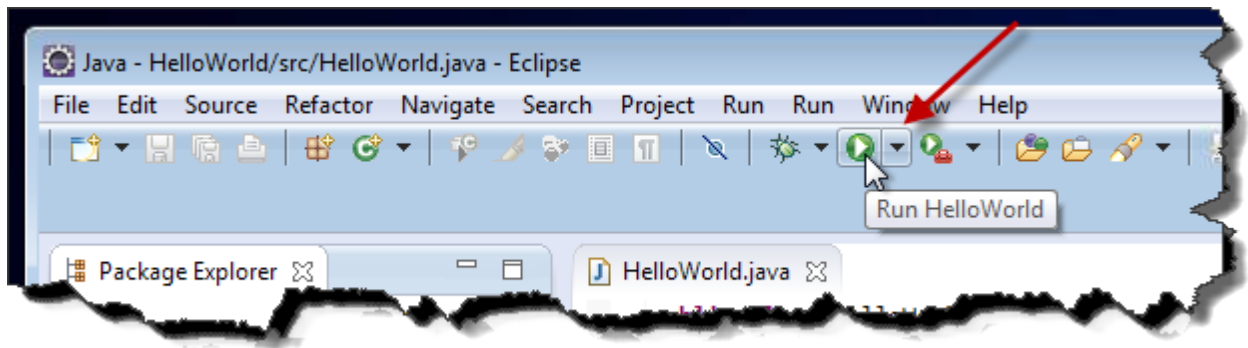
5.5 Execute HelloWorld. To run an application, right-click on the HelloWorld.java file and select **Run As > Java Application**.



5.5.1 Look for a new view, the Console view, to open near the bottom of the IDE to display your execution results.



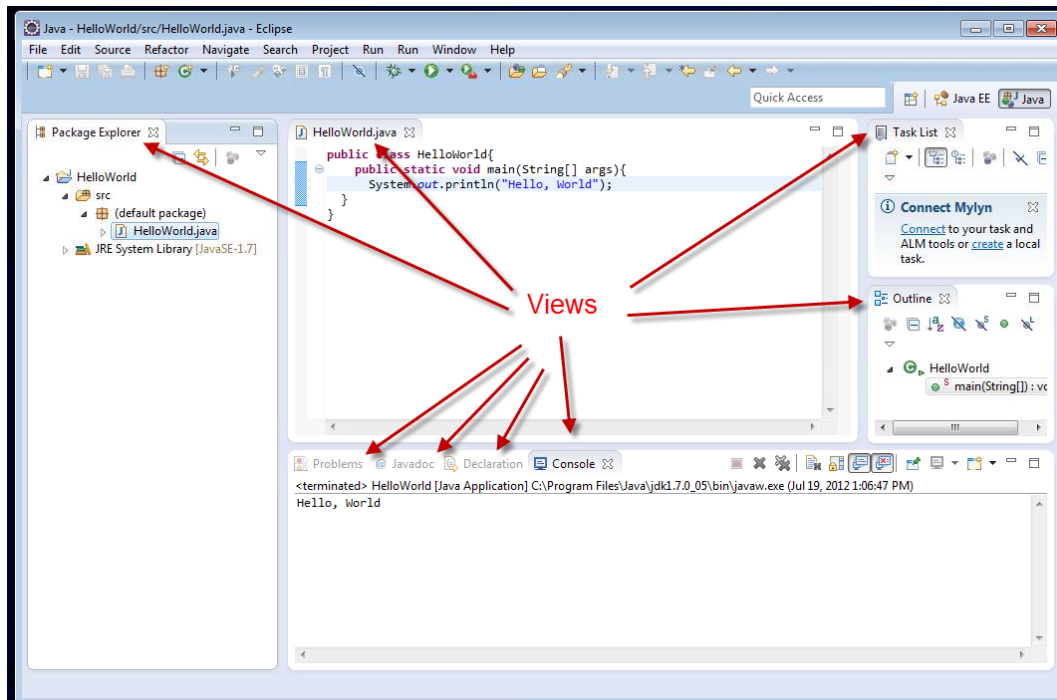
5.5.2 The next time you run your program, you can skip the menu selection and just click the green arrow run button.



The green arrow run icon will “rerun” whatever was run previously.

5.6 Eclipse is a large, powerful, and extensible IDE. There are a couple of basic Eclipse concepts or terms you should know when using it.

5.6.1 A view is one portion, tab, or frame of the overall display area. In the screen capture below, you can see many views, including the Console, Package Explorer, Task List, and Outline views, as well as the Java editor view. The extensive use of tab sets gives this editor many views. The Problems, Javadoc, and Declaration views are on the screen but not in the active tab set. Do you see these views?



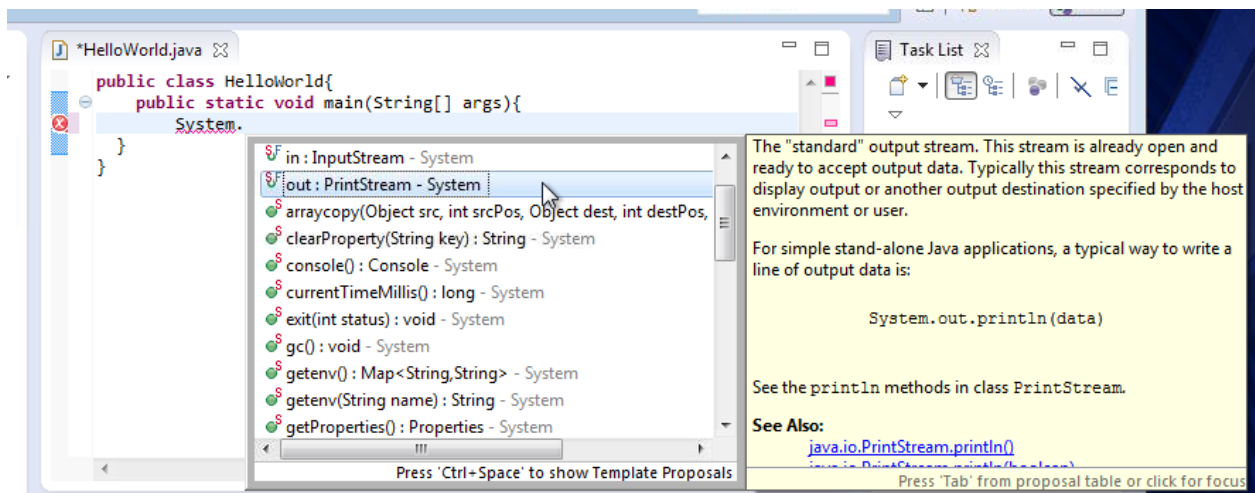
5.6.2 A perspective controls what views are actively shown at any time. Perspectives are meant to organize several widely used views for a given type of application development or developer task, such as debugging. There is a perspective for creating Web applications, one for Javascript work, one for debugging, and so on. By opening a new perspective, you might see completely different views. A tool bar in the upper right corner provides a quick way to switch perspectives.



Note: In some situations Eclipse automatically changes perspectives, for instance, when you start to debug code. The same files will be open, but views and many menus — context or right-click menus especially — will change.

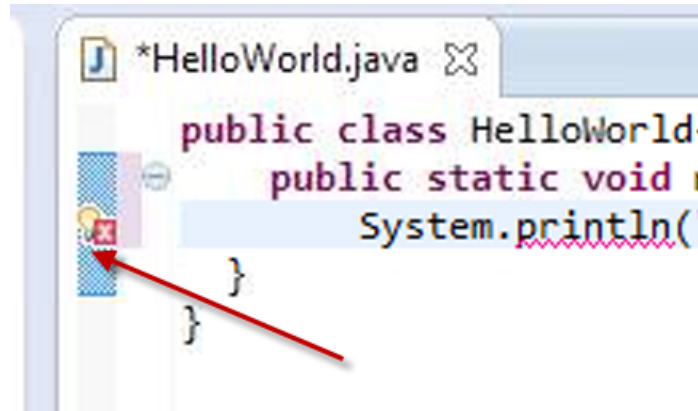
5.7 What features make Eclipse so attractive? Throughout class, you will be exposed to some of the features Eclipse has to offer. This lab cannot attempt to show you all the features of Eclipse, but here are a few very useful ones to check out.

5.7.1 Code assist. As you were typing your Java code in the editor, did you notice how the IDE tried to help you with your coding? For example, when you type a . (period) after a reference, Eclipse automatically shows you what method or data can be called on that reference.



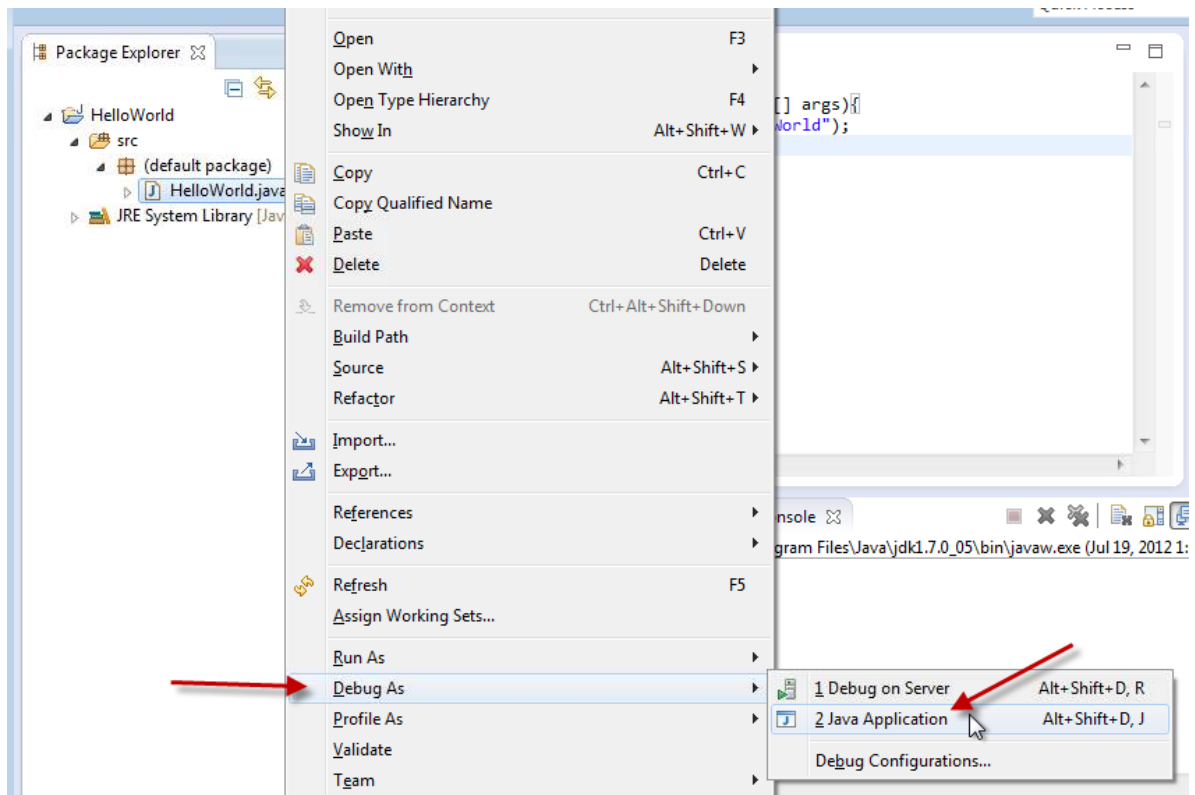
This can help reduce your dependence on exterior documentation. You can explicitly call for this code assistance help by pressing *Ctrl-space*. If your source code has many errors, the feature may temporarily stop working because it's easy to confuse a machine.

5.7.2 Eclipse will suggest changes to fix compiler errors in your code, using the light bulb icon to show you it has a suggested fix. To see what it suggests, simply use the mouse to click on the light bulb.

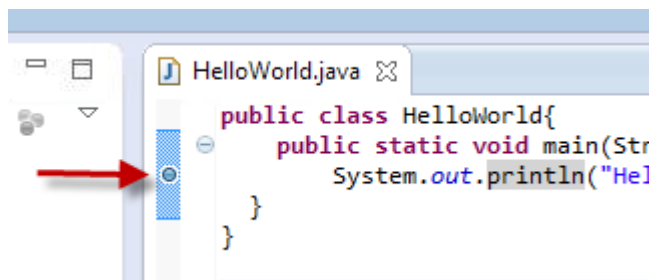


Note: Over time, you will see that Eclipse's suggested fixes aren't always on the mark, so don't always take the suggested fix without first looking at the proposed change. Thinking is still an integral part of your job.

5.7.3 Eclipse has a Debugger that can step through code. To run the debugger, choose Debug As instead of Run As from the right-click menu.



Before running the application in debug mode, a break point or stopping point must be established in the code. Set a break point by double-clicking on the left side of the editor, in the same space where errors and light bulbs appear. Break points appear as light blue dots (see below).





Step 6: Check out Eclipse

If you have any time left, please take some time to get used to Eclipse. This IDE will provide you with great support, but you have to get used to it. Call on your instructor for help with Eclipse if you have trouble finding what you need. Make sure you understand at a minimum the tasks listed below. Don't let the tool slow down your learning Java. If you find yourself frustrated by something Eclipse is doing, take a deep breath, relax, and ask a question.

6.1 Get these features down, and consider them required:

6.1.1 Creation of new classes

6.1.2 Compiling code — finding and fixing errors

6.1.3 Running your code

6.2 These are for optional work in this class but in the real world are quite necessary:

6.2.1 Debugging

6.2.2 Importing files



Lab Solution

HelloWorld.java

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



IDE Familiarization

Give Intertech a call at **1.800.866.9884** or visit Intertech's website.

