

## Getting Started With Java and Eclipse

---

### Sun's JDK

Sun's Java Development Kit (JDK) includes both a Java compiler and a Java interpreter (Java Runtime Environment, or JRE).

You can get the JDK from me or as a free download at <http://java.sun.com/javase/downloads/index.jsp>. Click on the Download link for JDK 6. You can also download the complete Java documentation set from that web page. The API documentation is especially useful.

I recommend that you download and learn how to use Eclipse, described below. However, you can compile and run Java programs using the JDK alone.

Note for Windows users: In order to be able to run the *javac* compiler from the command line, you must add the installation bin directory to your PATH environment variable. For example, by default jdk 6 installs into the directory C:\Program Files\Java\jdk1.6.0\. This installation directory contains a directory named bin. You would want to add this directory to your PATH:

```
C:\Program Files\Java\jdk1.6.0\bin
```

Next we will explain how to run the Java compiler and interpreter. Suppose that your Java program is in two files, Prog.java and Support.java. At the command line, type the command

```
javac Prog.java Support.java
```

If there are no syntax errors in the program, the compiler will compile these files, creating the files Prog.class and Support.class. Suppose that the **main** method you want to run is in Prog.java. This command will run it:

```
java Prog
```

Note that you type the name of the class containing the **main** method, but you do *not* type the .class file extension.

### What Is Eclipse?

Eclipse is an open source, extensible *Integrated Development Environment (IDE)* that is very popular among Java programmers. It runs on most versions of Windows, Red Hat and SuSE Linux, Solaris, Macintosh OS X and other platforms. It has a Java editor that finds all syntax errors as you type. It has a good source-level debugger. Plus, it's free.

## Getting and Installing Eclipse

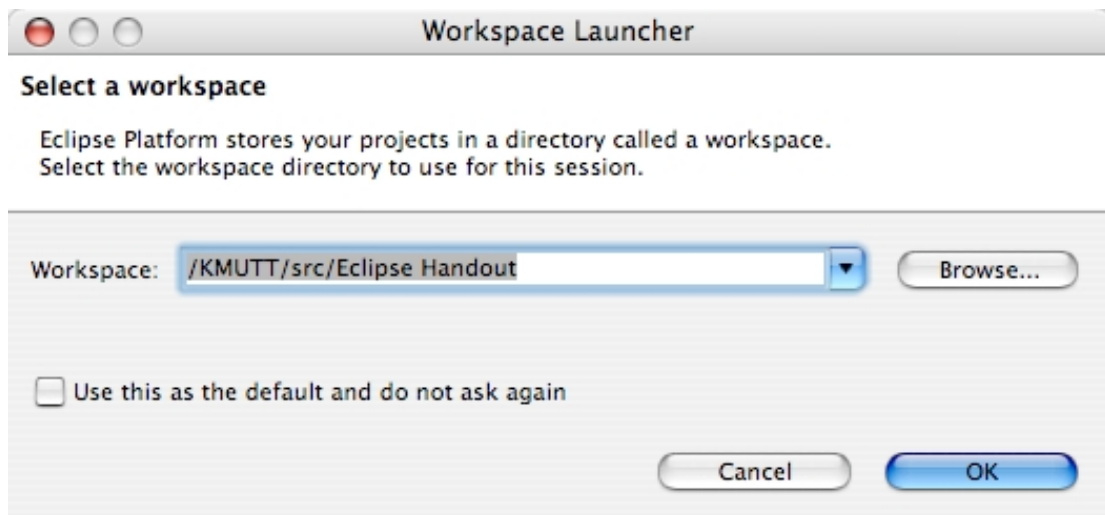
You can get Eclipse from me or you can download it at <http://www.eclipse.org/downloads/>. Eclipse is written in Java, so before you can run it, you must already have installed a JDK or JRE (Java interpreter) on your computer.

For MS Windows, the Eclipse download is a big Zip file. Unzip Eclipse and put it wherever you like.

Eclipse stores all of your projects and source files in a workspace directory of your choosing. Before you run Eclipse for the first time, create the directory that you will use for your programming projects.

## Starting Eclipse

When you launch Eclipse, you will see<sup>1</sup>:



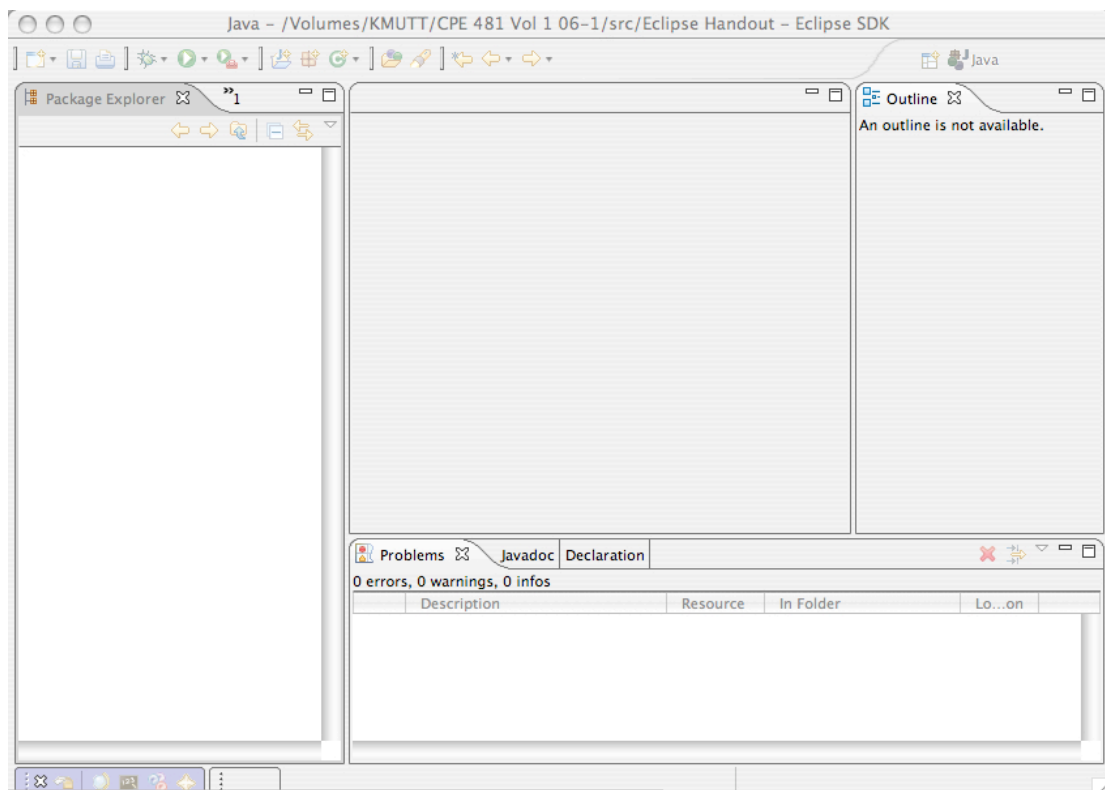
Click Browse... and navigate to the folder you have created to hold your workspace, then click OK. You will see:

---

<sup>1</sup> Note: the screenshots for this handout were prepared on a Macintosh. They will look a little different on other platforms or with different versions of Eclipse.



The icons on the screen center give you access to the Eclipse documentation. The arrow icon at the right will take you to your “workbench”. Click on it to bring up your working environment:



There isn't much in the workspace yet.

## Perspectives and Views

When you program, you use different tools at different times. When you are editing, you need a list of files in your project. When you are debugging, you need windows for viewing variable values. When you are designing, you need inspectors to browse the class structure of your program. If you keep open all windows for all of these tools, your monitor will become very cluttered.

Eclipse solves this problem by using *perspectives*. A perspective is a set of tool windows. Initially, you are in the Java perspective. This perspective has tools for writing and executing Java code. There are also specialized perspectives for Debug, Java Browsing and Java Type Hierarchy.

All perspectives share the same editors. Whatever files you are currently editing, switching perspectives switches only the tools that surround the editing windows.

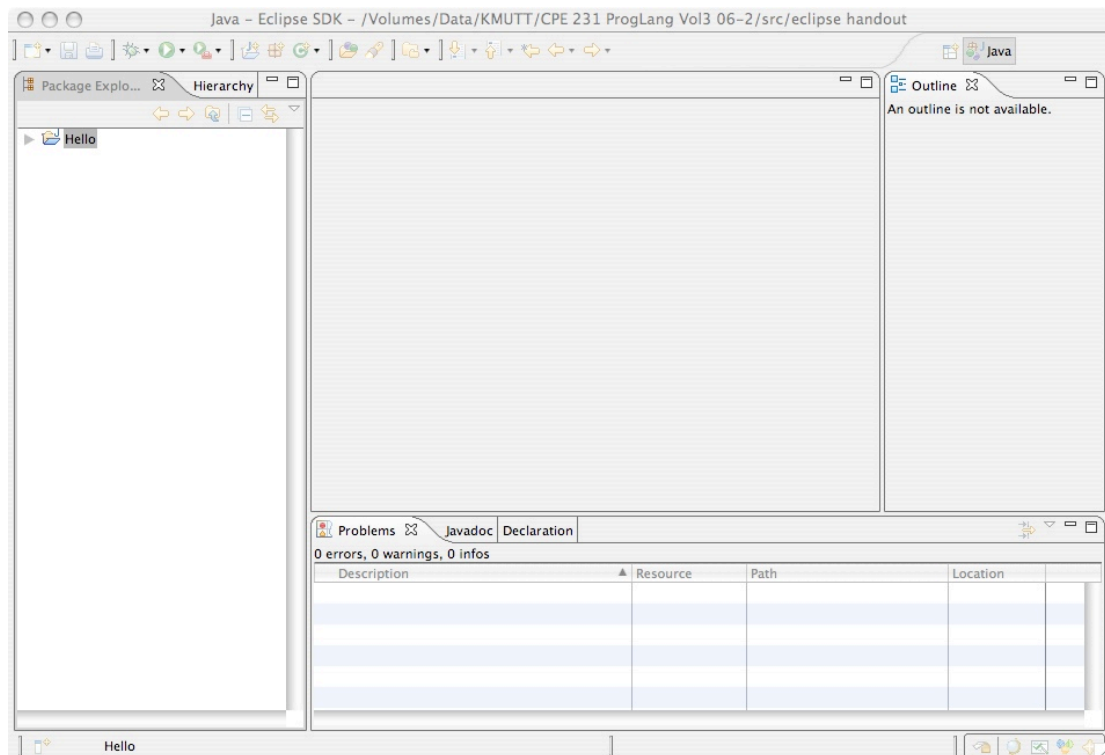
A *view* represents some tool for working with your project. In the above window the Package Explorer view lists all of your projects and their contents, organized by Java Package.

## Create a “Hello, World” Program

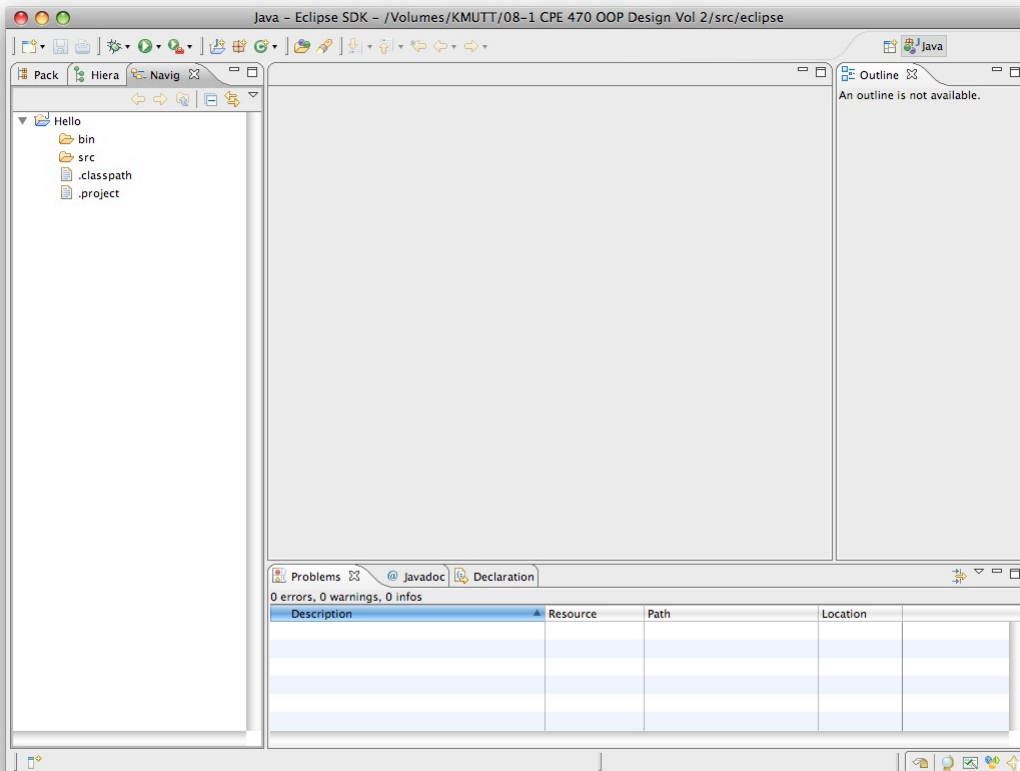
We will go through the steps required to create and run a simple Java program.

1. Create a Java a project.
  - a) Choose the menu item File/New/Project... .
  - b) Select Java Project in the category list. Click Next.
  - c) Give the project a name (“Hello”, for example). Click Finish.

You will see:

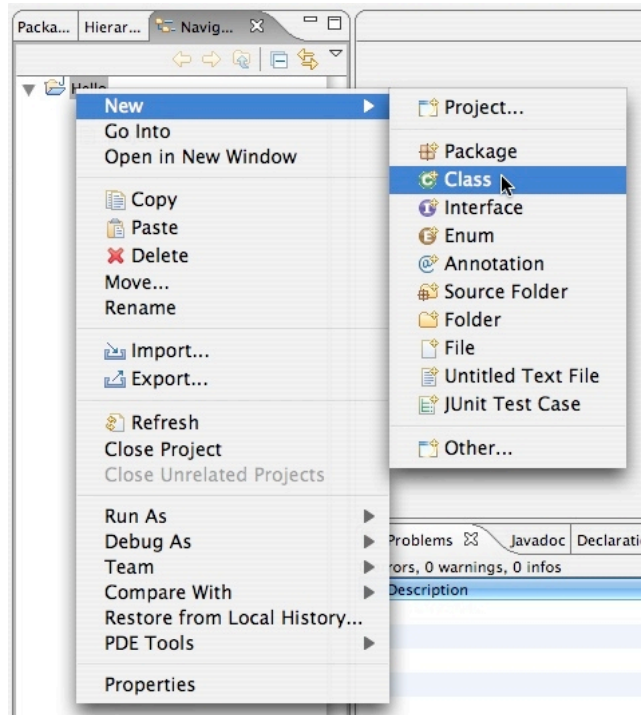


The Package view shows your project classes organized by package. We won't be using packages for this simple program so that view is not especially useful. The Navigator view would be most useful but it is not shown in this perspective. You can add any view you like to any perspective. Choose the menu item Window/Show View/Navigator. You will see:



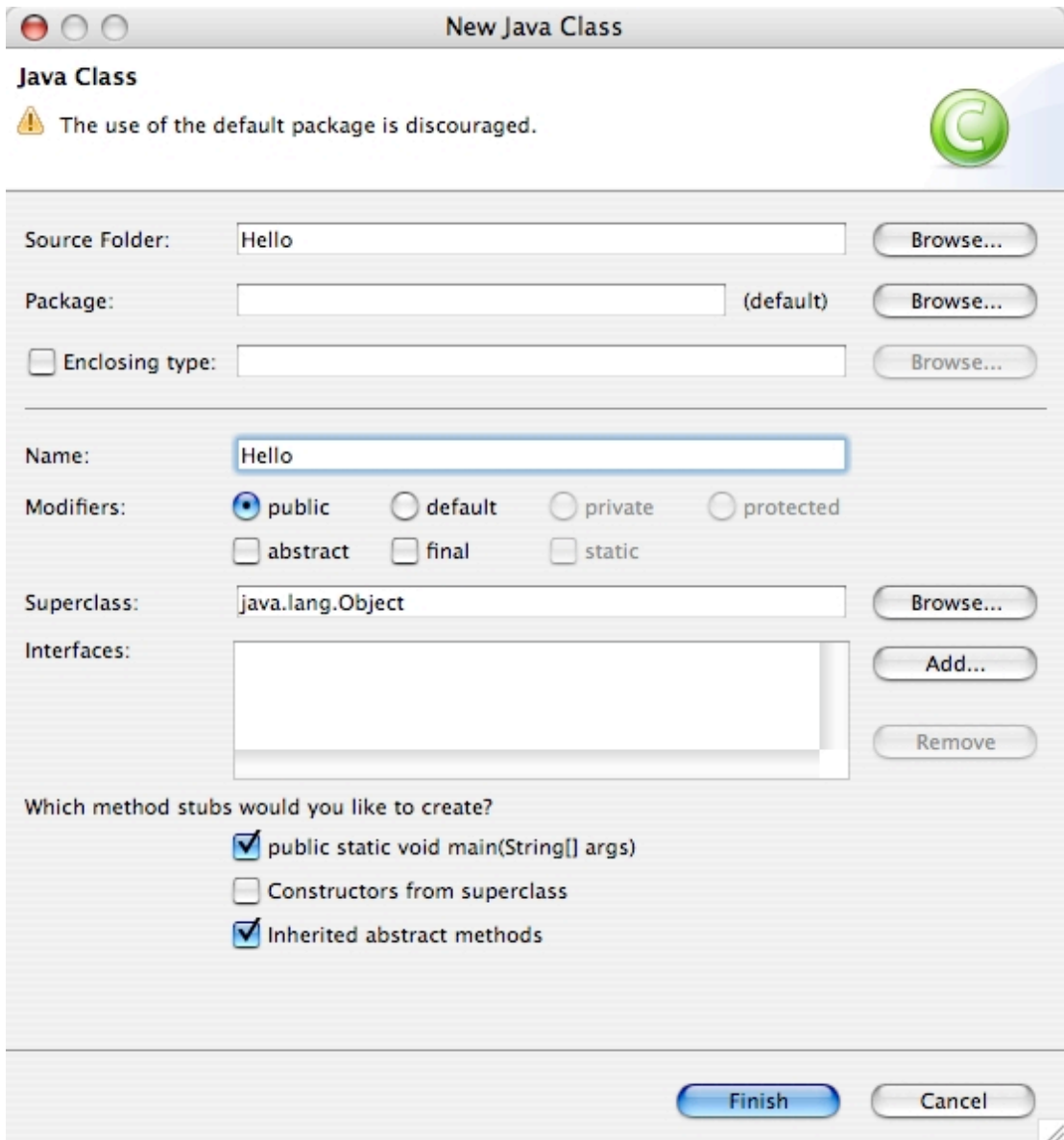
The Navigator view has been added. A perspective can contain many views, so, to save screen space, views are organized into *frames* that may contain multiple *tabs*. The Navigator and Package views both appear in the same frame, but they each have their own tab. See also, in the bottom frame, that there are separate tabs for the Problems, Javadoc and Declaration views. When you edit or run a program, any compilation errors will appear in the Problems view. Notice that the Hello project lists two files, .classpath and .project. These files are created automatically by Eclipse for each project. The folder src will hold all .java source files for the project and the folder bin will contain the compiled .class files.

2. Create a new Java class:
  - a) You can select the menu item File/New/Class. However, it is time to learn another way of selecting commands. Eclipse is a large program with many menus and commands. Sometimes it is difficult to remember which menu has the command you need. Eclipse relies heavily on “context sensitive popup menus.” If you select something and right-click on it (click your secondary mouse button, or, on a Macintosh, <ctrl>+click), a menu pops up containing only those commands that are relevant to the selected item. Try it. Right-click on the Hello project:



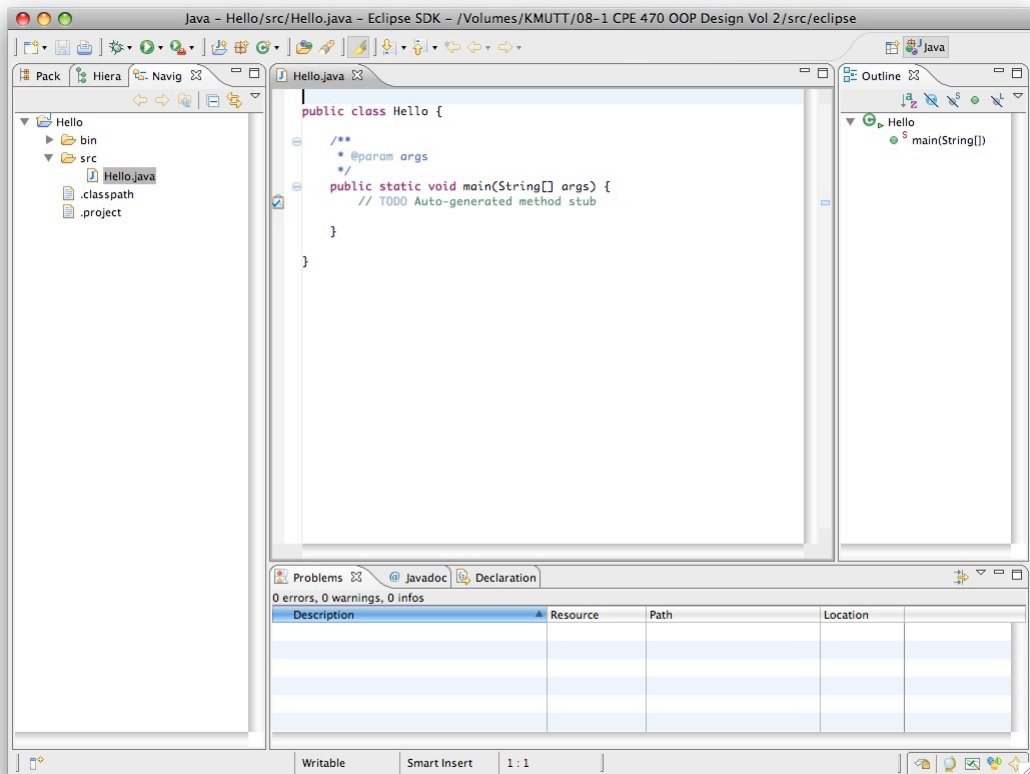
You can see that the popup menu has all commands that can be applied to the Hello project. This is true in general with Eclipse. When in doubt, look at the popup menu for any item you want to work with. Here, select New/Class.

- b) In the dialog, name the class Hello, and check the box that asks Eclipse to create **public static void main(String[] args)**. Ignore the warning that “The use of the default package is discouraged.” Click Finish:





You will see:



You can see that `Hello.java` has been added to the project. If you open the `bin` directory, you will see that `Hello.class` has been added to it. Eclipse automatically compiles each source file every time you save it. `Hello.java` has also been opened in an editor window. You can edit as many files as you wish; each file will have its own tab in the editor pane. Note that Eclipse has filled in some of the code: some comments, the class declaration, and the **main** method.

Note also that the Outline view now shows the **Hello** class and its contents. The outline will list every variable and method in the class. Different icons distinguish constructors and static fields or methods. Color coding distinguishes public, private and protected items. You can see your whole class at a glance. If you click on something in the outline, the editor will scroll to it. This is true of many of the views. If you select or double-click on something, an editor is opened or some other useful action occurs.

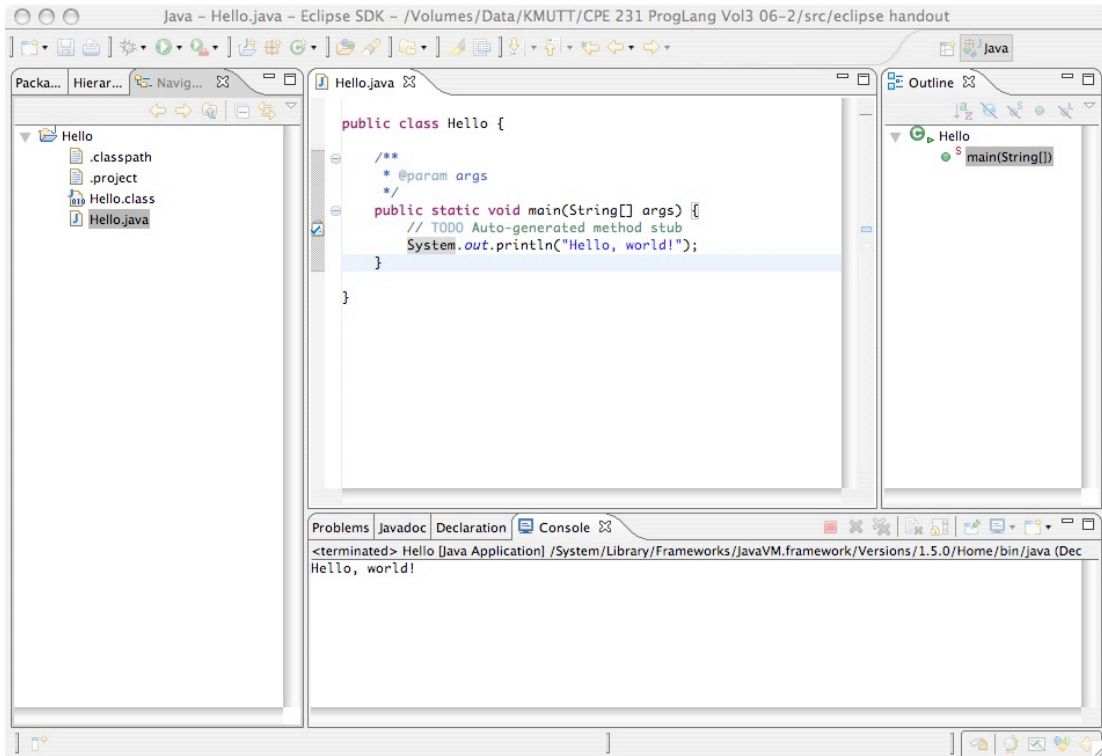
c) To finish the class, add this line to **main**:

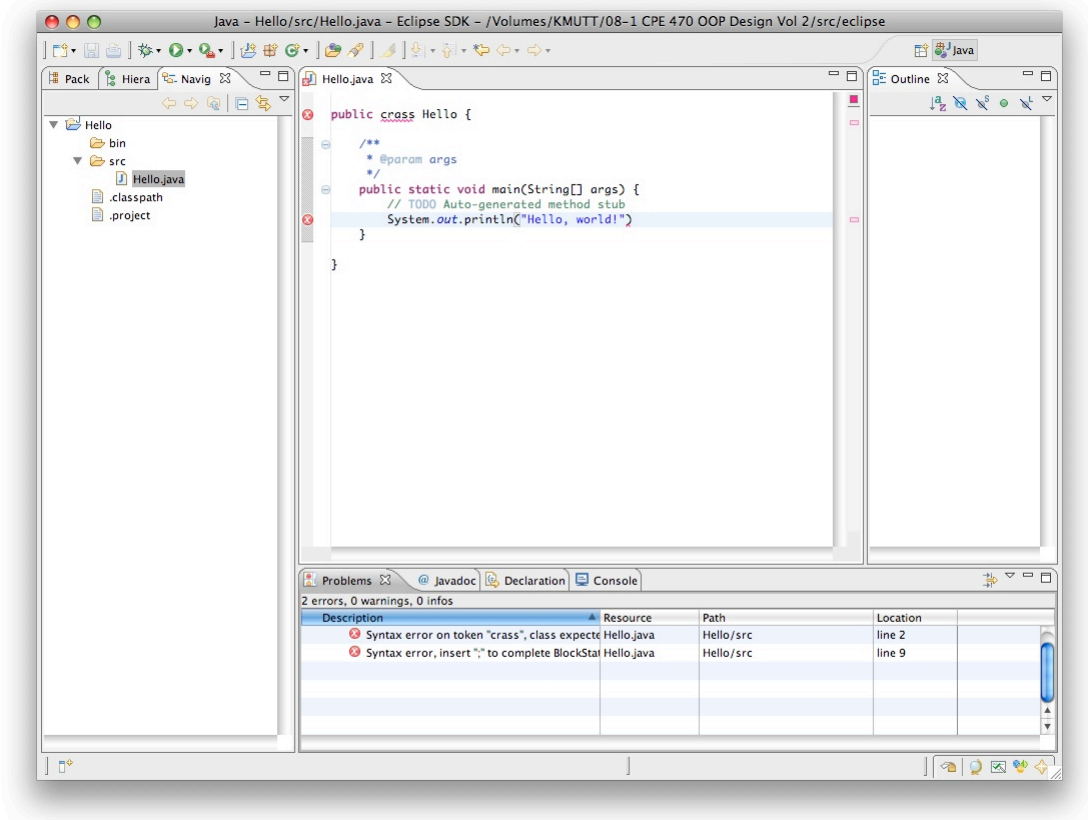
```
System.out.println("Hello, world!");
```

The editor automatically indents and formats your code as you type. It adds closing parentheses and braces for you.

Save your work. You can do this by clicking on the floppy disk icon on the toolbar or by choosing File/Save.

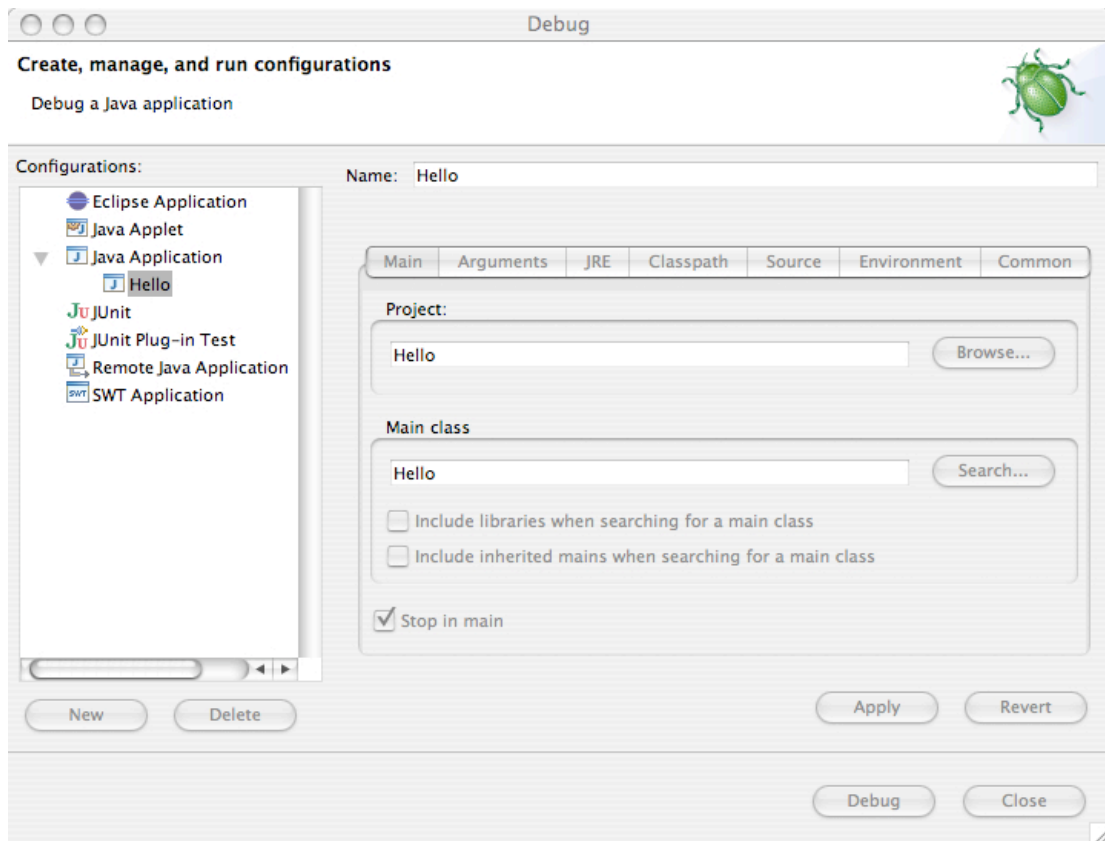
3. Run your program. Select Hello.java in the Navigator view. In its popup menu, choose Run As/Java Application. If you haven't saved your files, you will be asked to do so, then the program runs. The bottom frame shows a Console view. The Console is the window for text I/O:



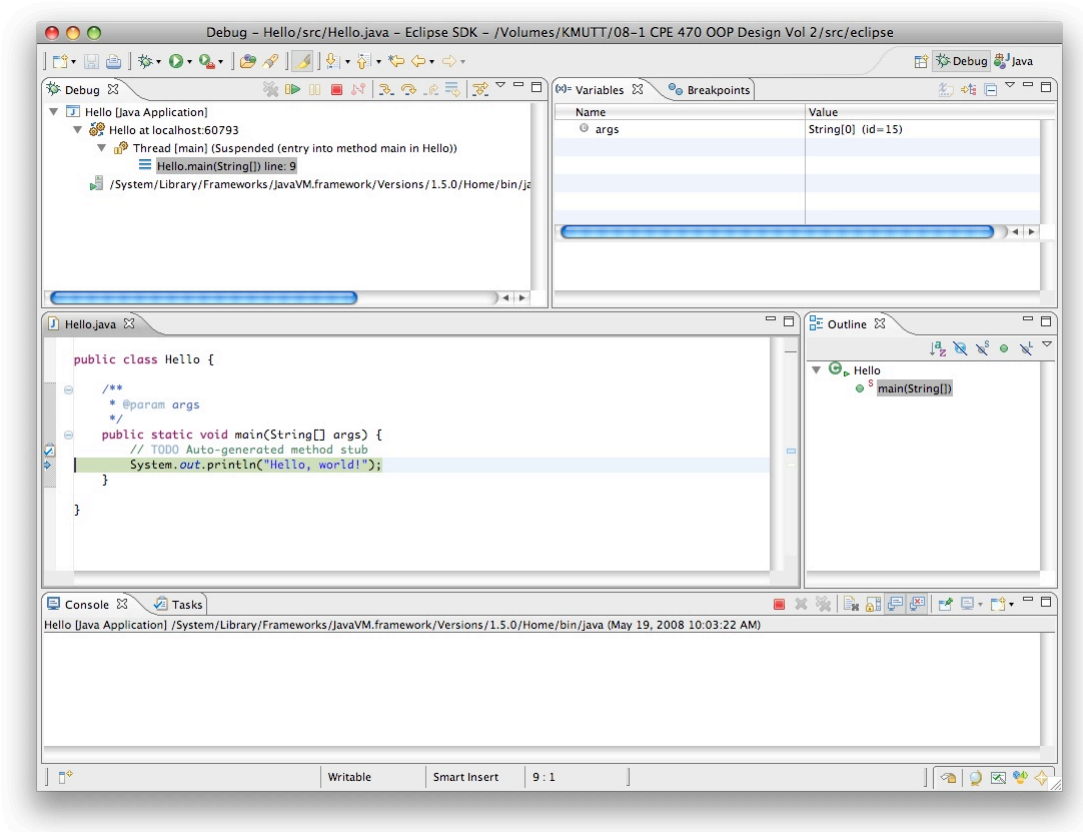


## Debugging

Eclipse has a good source-level debugger. The first time you want to debug a program, select the class with that program's **main** method in the Navigator view and choose Debug As/Open Debug Dialog... from its popup menu:



There are many options you can set here, but the important one is to check the box that says Stop in main. Then, click Debug. You will get an alert asking if it's okay to switch to the Debug perspective; answer Yes. You will see:



All perspectives are listed at the top-right corner of the window. You can switch back and forth between the Debug and Java perspectives simply by clicking on their names.

Execution has halted at the first statement of **main**. This program is not very interesting to debug, but the debugger is useful for larger programs. You can single-step (execute one statement at a time), run until a breakpoint (any line that you mark as a breakpoint), and examine the value of any variable or expression. All accessible variables are shown in the Variables tab. You can also add an Expressions view tab that lets you type in and evaluate any expression.

The main debugger commands from the Run menu (and their function key shortcuts) are:

**Step Into (F5):** Execute one statement and then break. If the statement is a function call, stop at the first line of the function implementation.

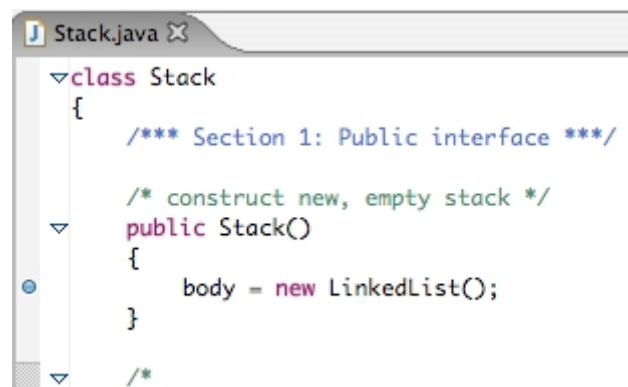
**Step Over (F6):** Execute one statement and then break. If the statement is a function call, execute the entire function call as if it were a single statement and break when the function returns.

**Step Return (F7):** Execute until the current function returns, then break.

**Resume (F8):** Execute until the next breakpoint or until the program exits.

**Terminate:** Cancel program execution.

You can set a breakpoint at any line in your source code. Put the cursor anywhere in the line and choose Run/Toggle Line Breakpoint, or just double-click in the gray area at the left of the line. A blue dot will appear at the left of the line. Here is an example from a different project:



```
Stack.java
class Stack
{
    /** Section 1: Public interface */
    /* construct new, empty stack */
    public Stack()
    {
        body = new LinkedList();
    }
    /*
```

## Help

The Help menu will take you to complete Eclipse documentation. Choosing Help/Welcome takes you back to the screen you saw when you first launched Eclipse.

## Moving Files Around

Each project listed in the Navigator window is stored as a folder (directory) in your workspace folder. You can edit any files in the folder using any editor you like. If you add files created elsewhere to a project folder, Eclipse won't know that they are there. If you select the project and choose Refresh from the popup menu, Eclipse will update its display to reflect all the files you have added or deleted from the project folder.

If you delete a file from a project using Eclipse, that file is also deleted from its folder. If you delete a project, Eclipse asks you whether you also want to delete the folder. If you save the folder, you can bring that project back into your workspace later by using the File/Import command. You can also use the Import function to copy any external files into a project folder; this is an alternative to manually copying files into the project folder. There is also an Export function.

## Source and Refactor

The Source and Refactor menus have many commands to help with editing and revising code. For example, if you select a block of text, there are commands to reformat it or fix its indentation, to create comments, to put the code inside a **try/catch** block, *etc.* If you use these tools to rename a class or variable, all references to the old name are updated automatically.