Introduction to Eclipse

Overview

- Eclipse Background
- Obtaining and Installing Eclipse
- Creating a Workspaces / Projects
- Creating Classes
- Compiling and Running Code
- Debugging Code
- Sampling of Features
- Summary

What is Eclipse?

- Eclipse started as a proprietary IBM product (IBM Visual age for Smalltalk/Java)
  - Embracing the open source model IBM opened the product up
- Open Source
  - It is a general purpose open platform that facilitates and encourages the development of third party plug-ins
- Best known as an Integrated Development Environment (IDE)
  - Provides tools for coding, building, running and debugging applications
- Originally designed for Java, now supports many other languages
  - Good support for C, C++
  - Python, PHP, Ruby, etc…

Prerequisites for Running Eclipse

- Eclipse is written in Java and will thus need an installed JRE or JDK in which to execute
  - JDK recommended

Eclipse on GL

- This years coordinated release (known as Ganymede) of the Eclipse IDE for Java Developers has been installed on GL
  - From any of the Linux machines in the labs simply run the command **eclipse**

Obtaining Eclipse

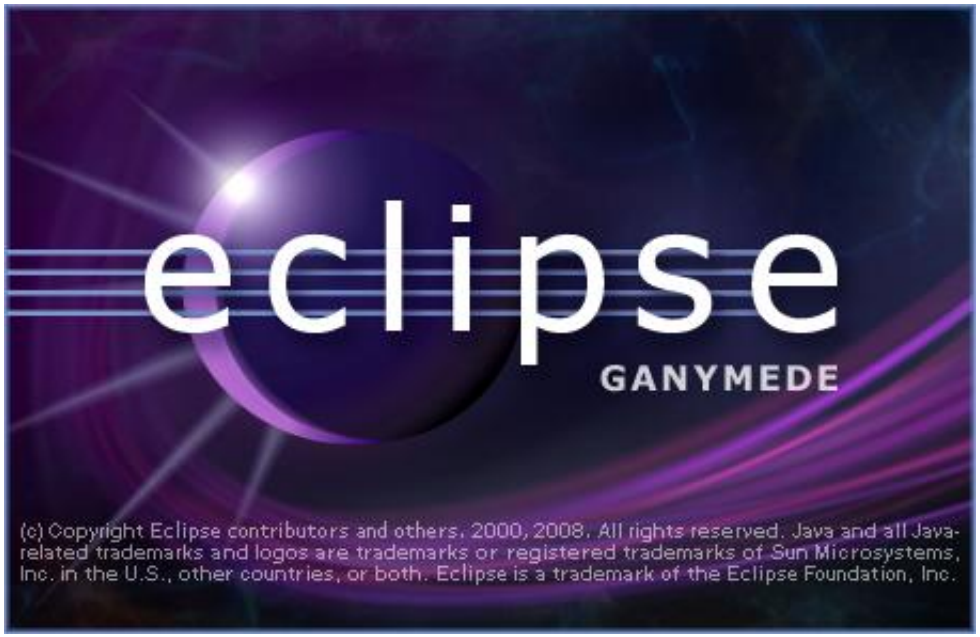
- Eclipse can be downloaded from…
  - http://www.eclipse.org/downloads/packages/
  - Be sure to grab “Eclipse IDE for Java Developers”
- Eclipse comes bundled as a zip file (Windows) or a tarball (all other operating systems)
  - Some versions of Linux (i.e. Fedora, Ubuntu) offer Eclipse in their respective repositories and can be downloaded using the appropriate tool (i.e. yum, apt-get)

Installing Eclipse

- Simply unwrap the zip file to some directory where you want to store the executables
- On windows
  - I typically unwrap the zip file to C:\eclipse\
  - I then typically create a shortcut on my desktop to the eclipse executable
    - C:\eclipse\eclipse.exe
- Under Linux
  - I typically unwrap to /opt/eclipse/

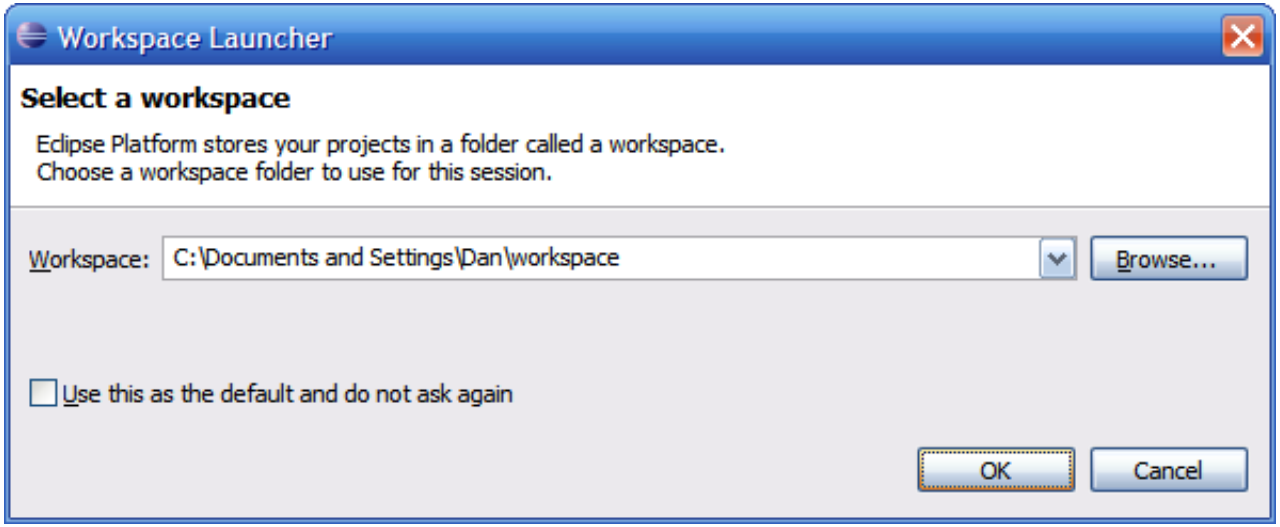
Launching Eclipse

- Once you have the environment setup, go ahead and launch eclipse
- You should see the following splash screen…



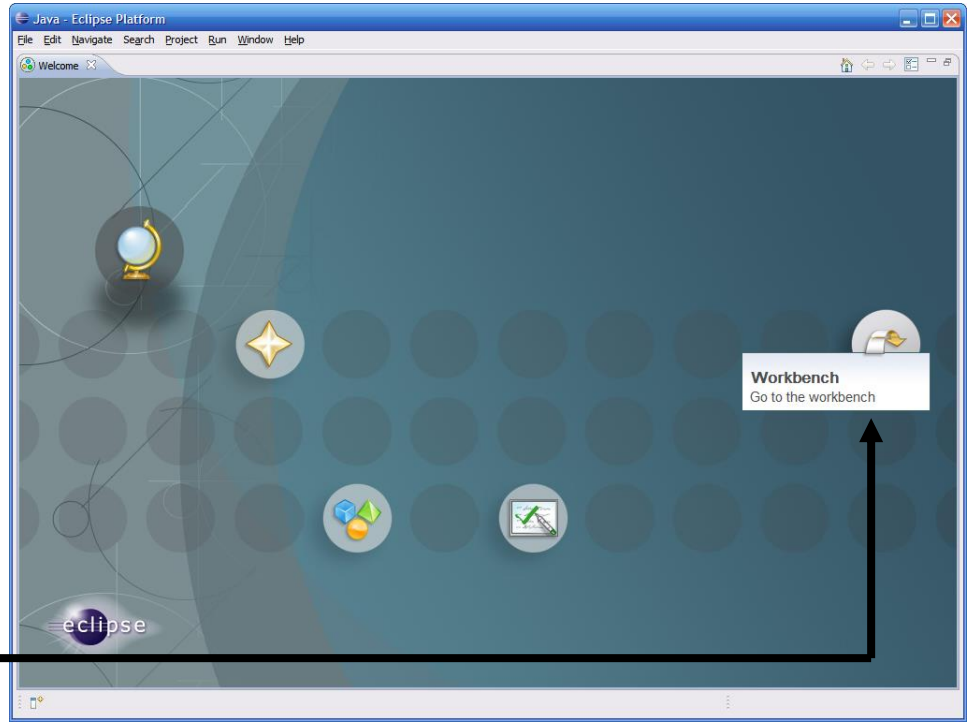
Selecting a Workspace

- In Eclipse, all of your code will live under a workspace
- A workspace is nothing more than a location where we will store our source code and where Eclipse will write out our preferences
- Eclipse allows you to have multiple workspaces – each tailored in its own way
- Choose a location where you want to store your files, then click OK

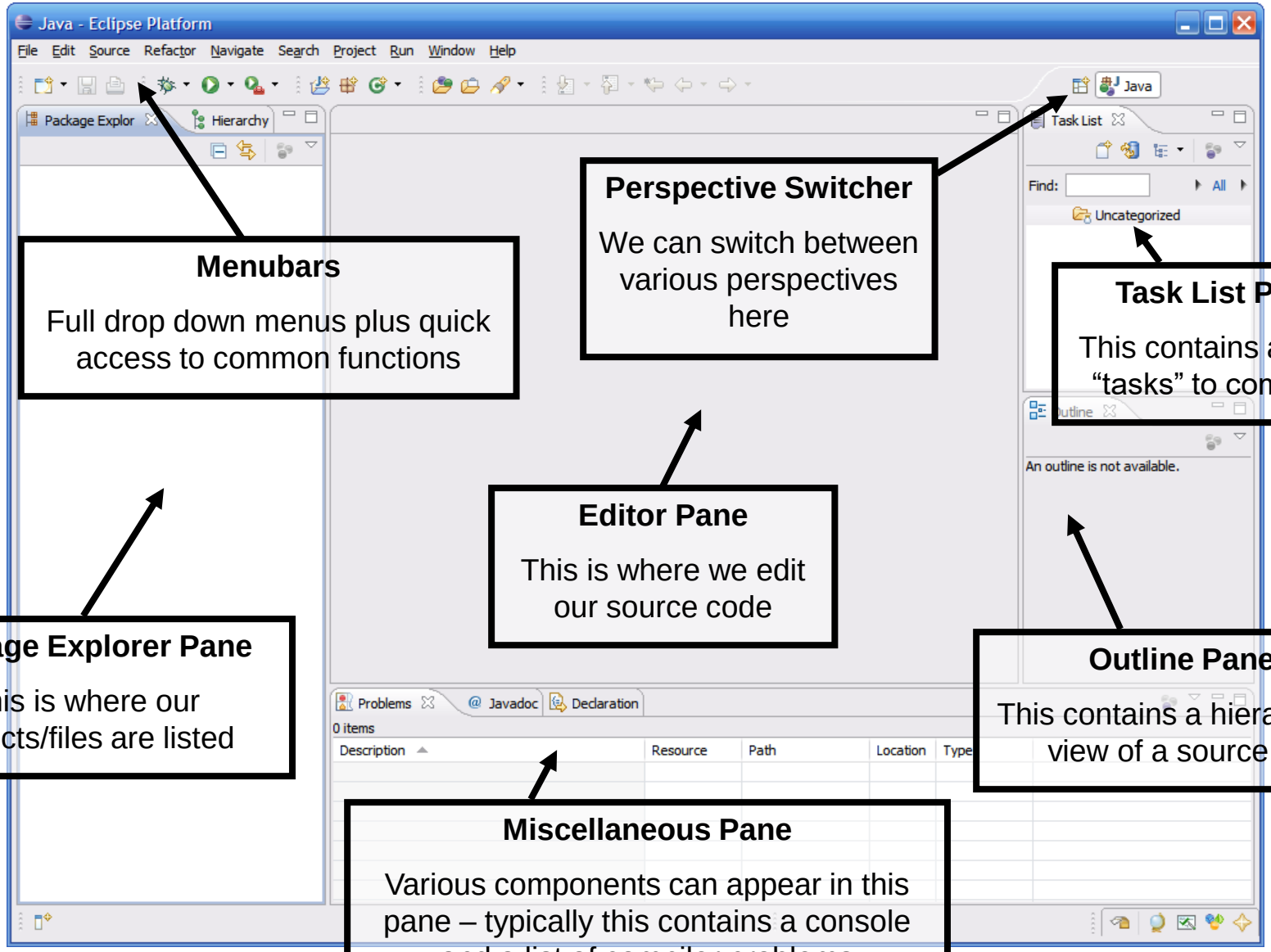


Welcome to Eclipse

- The first time you launch Eclipse, you will be presented with a welcome screen
- From here you can access an overview to the platform, tutorials, sample code, etc…
- Click on the arrow on the right to get to the actual IDE

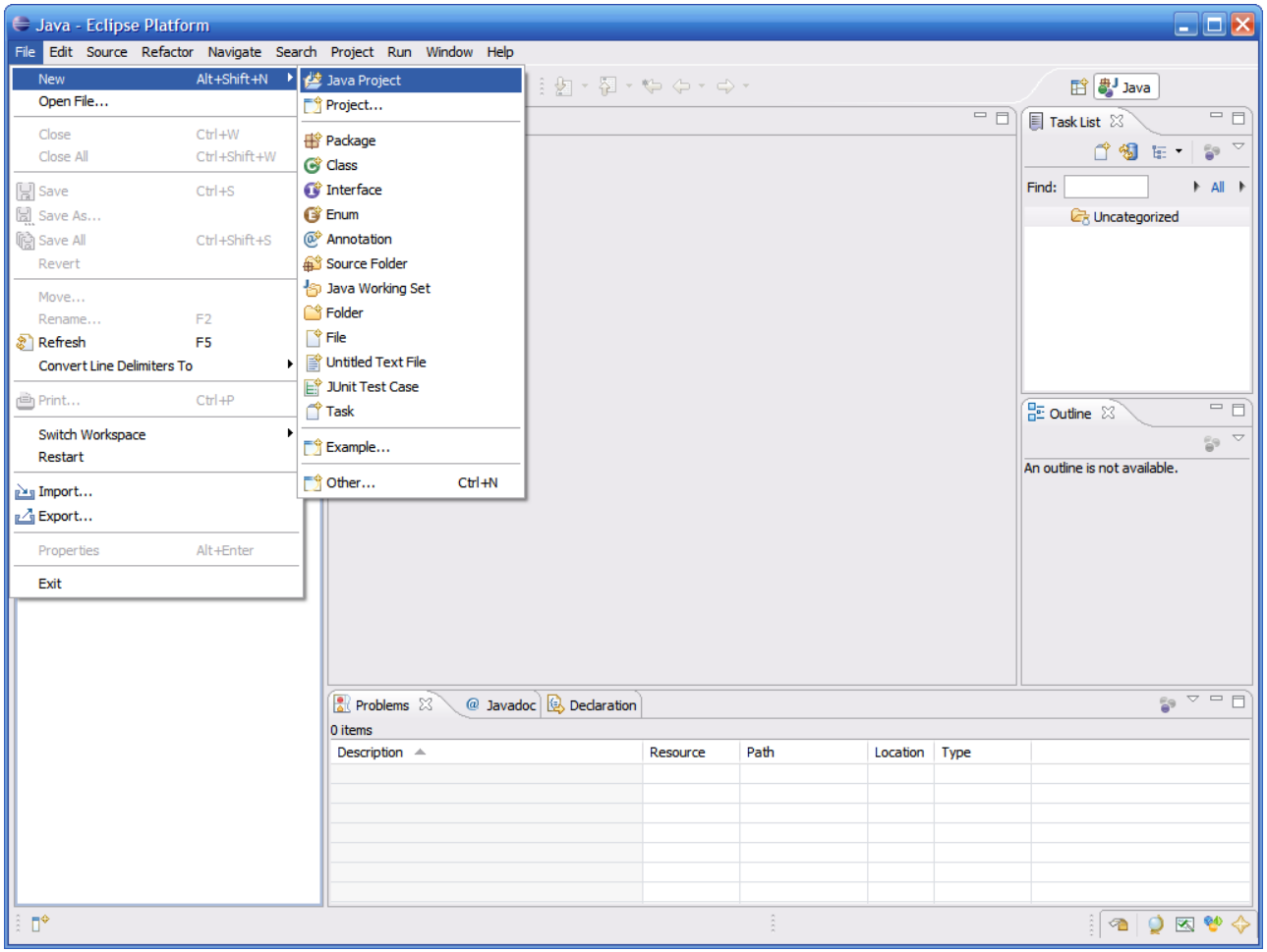


Eclipse IDE Components

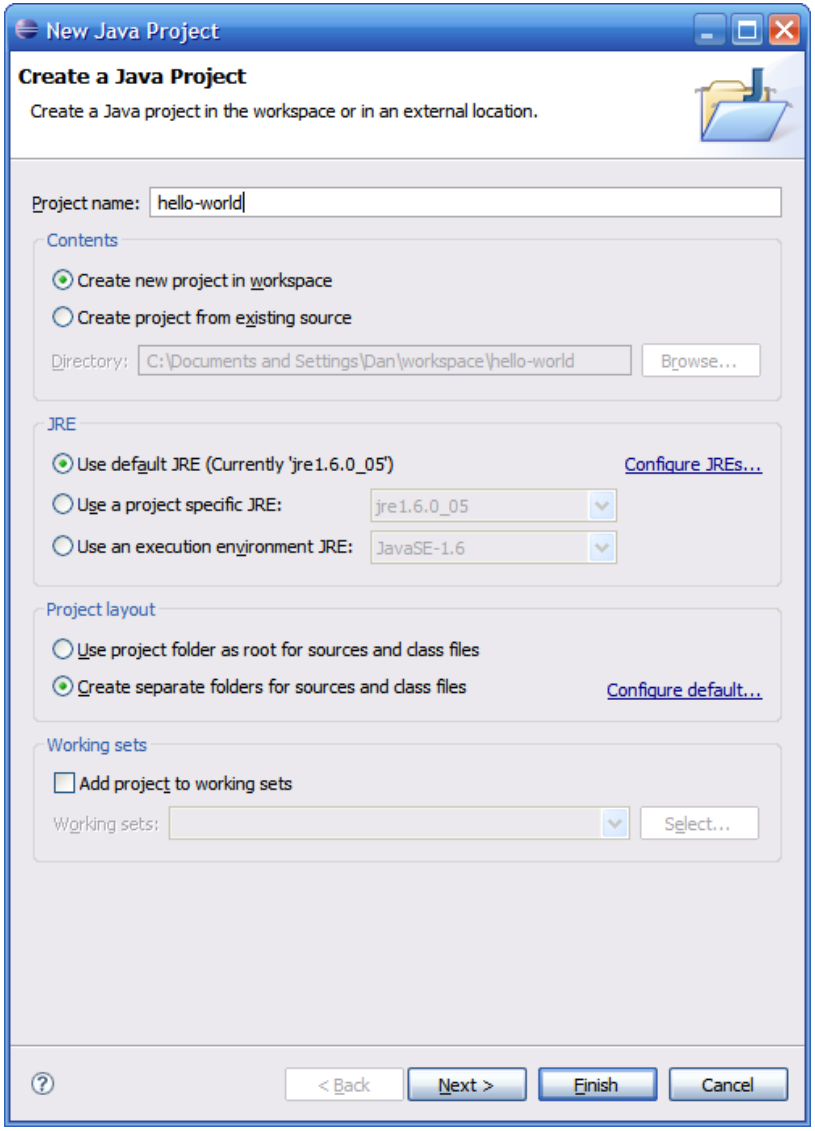


Creating a New Project

- All code in Eclipse needs to live under a project
- To create a project: File → New → Java Project



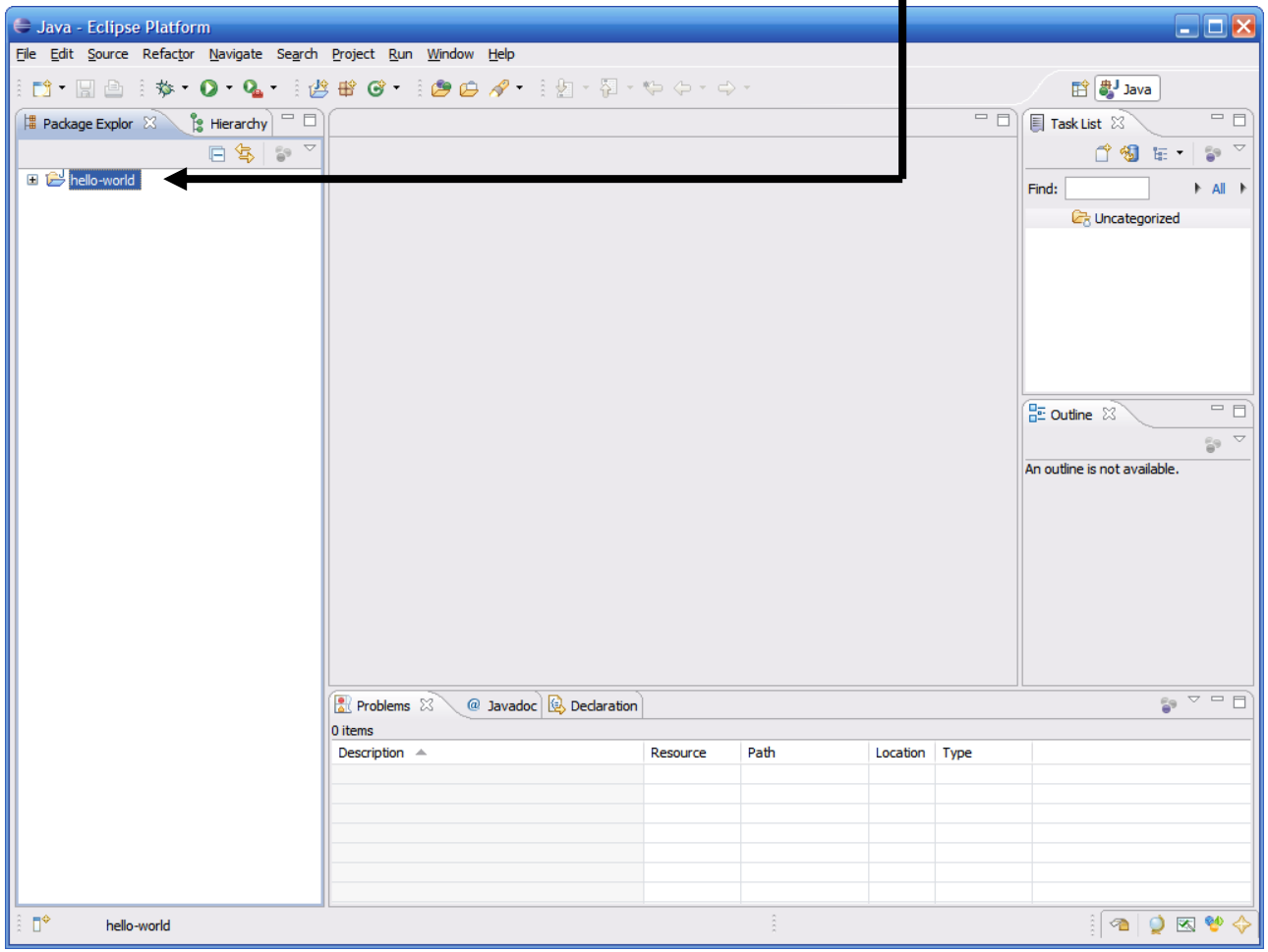
Creating a New Project (continued)



- Enter a name for the project, then click Finish

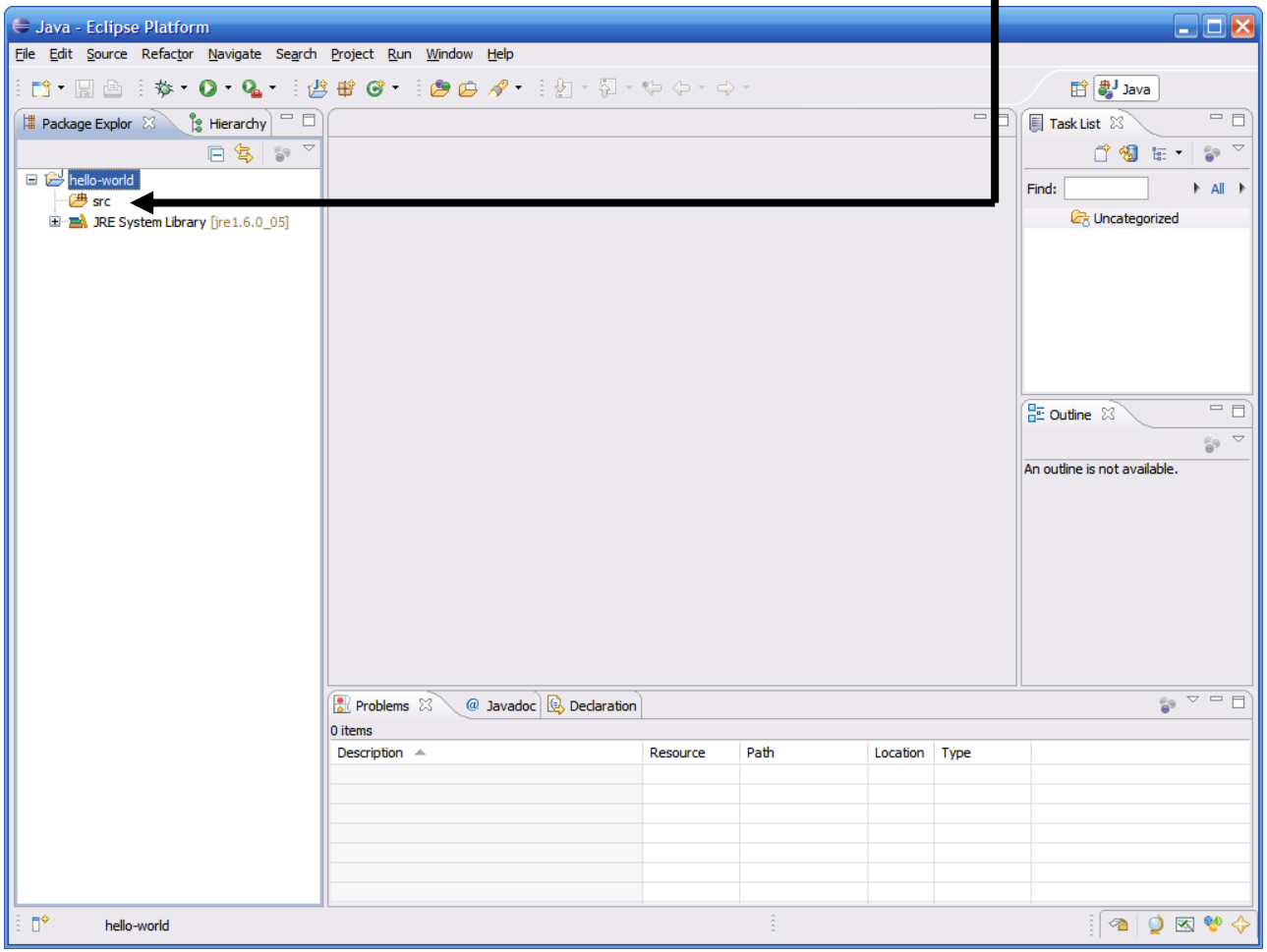
Creating a New Project (continued)

- The newly created project should then appear under the Package Explorer



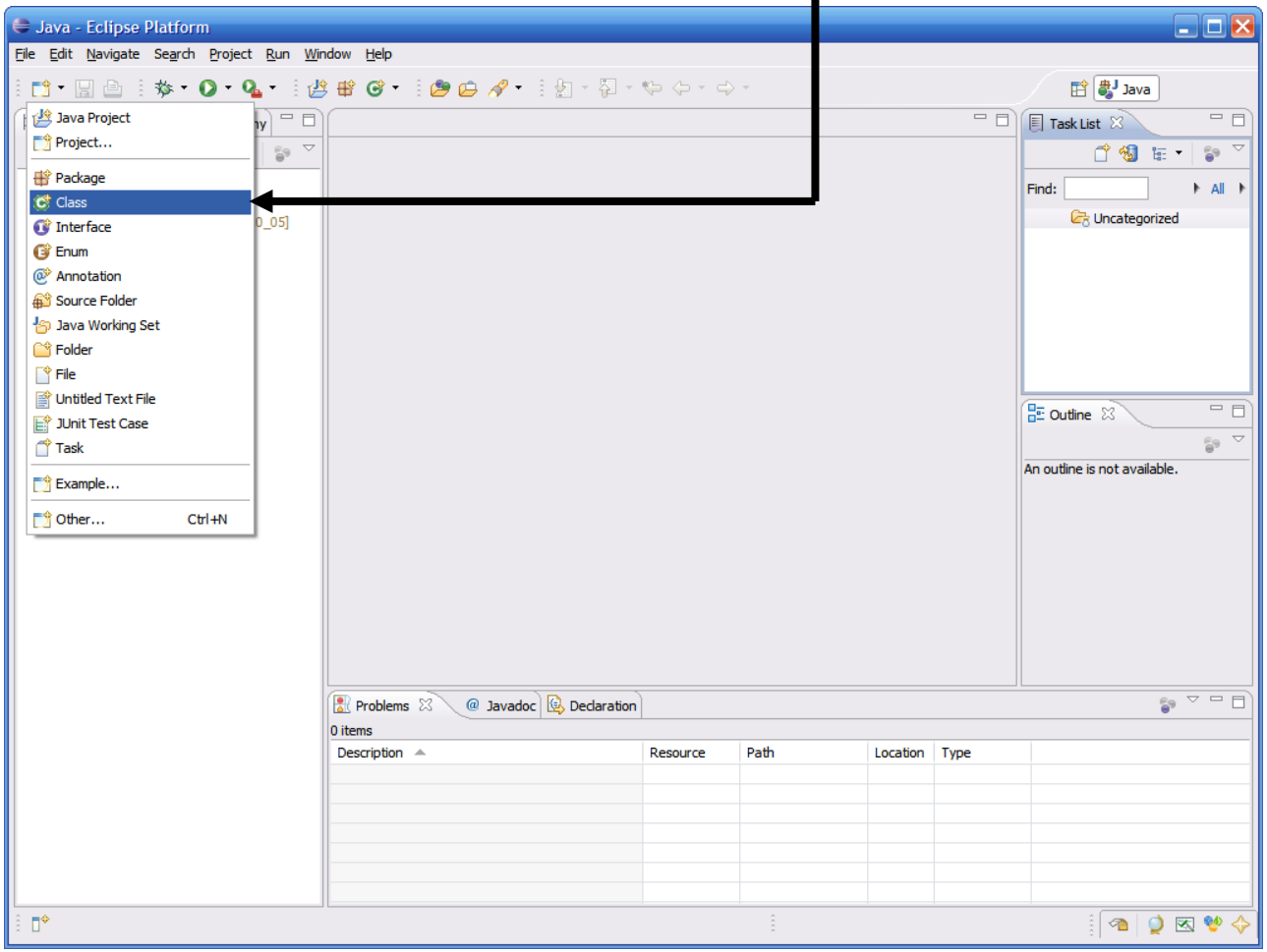
The src folder

- Eclipse automatically creates a folder to store your source code in called src

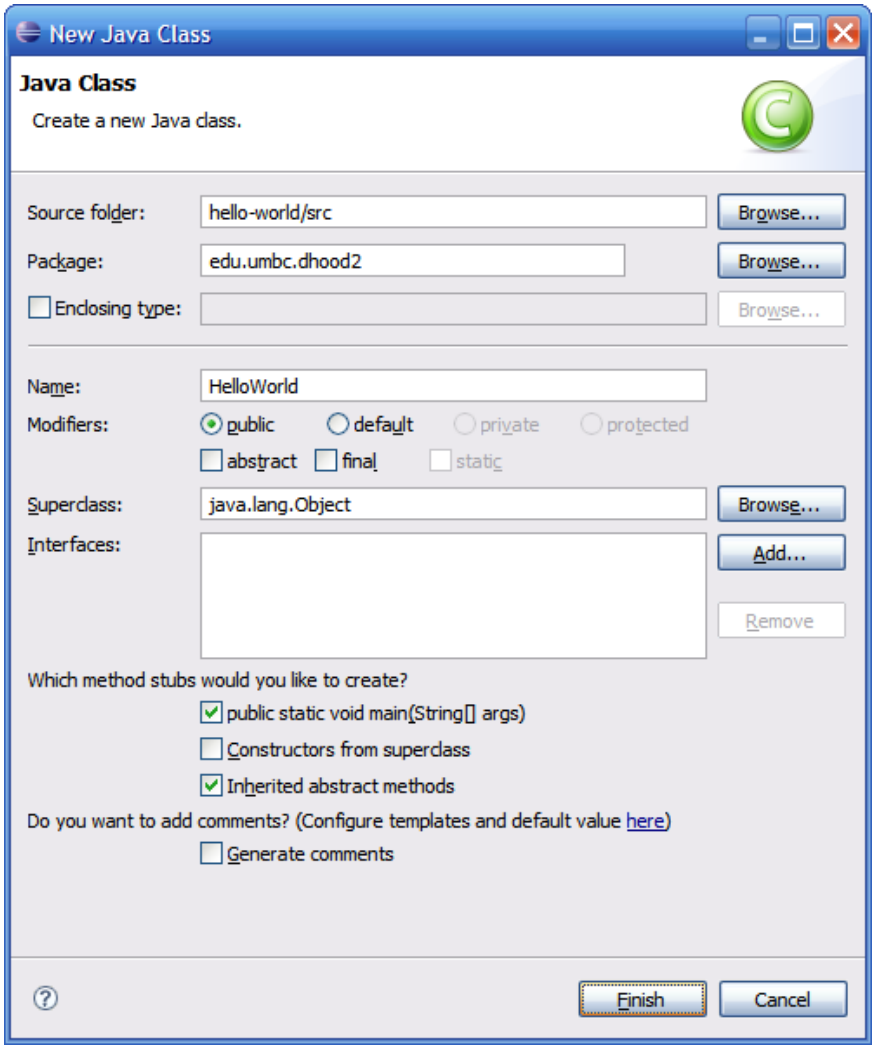


Creating a Class

- To create a class, simply click on the New button, then select Class



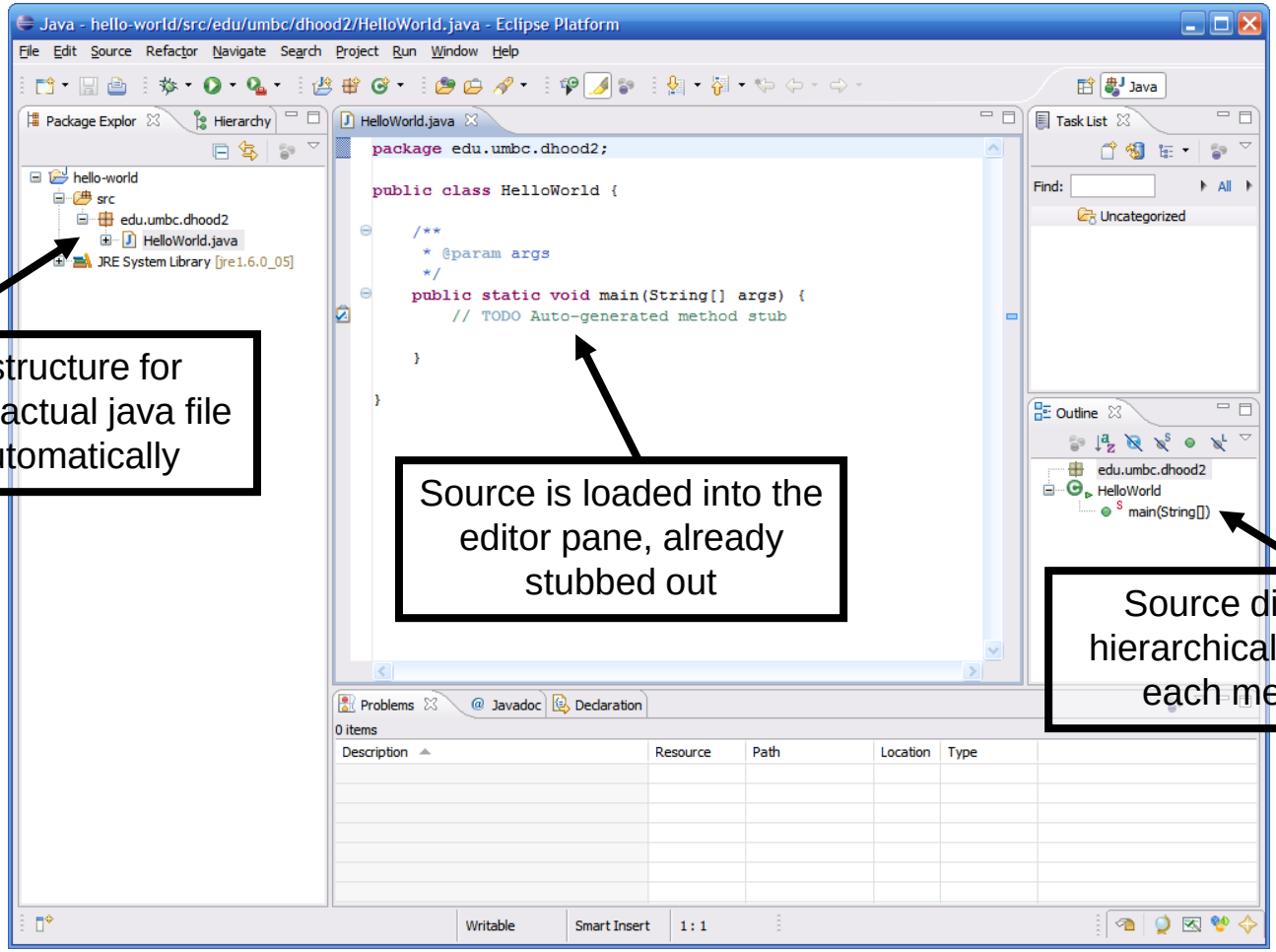
Creating a Class (continued)



- This brings up the new class wizard
- From here you can specify the following...
  - Package
  - Class name
  - Superclass
  - Whether or not to include a main
  - Etc…
- Fill in necessary information then click Finish to continue

The Created Class

- As you can see a number of things have now happened…

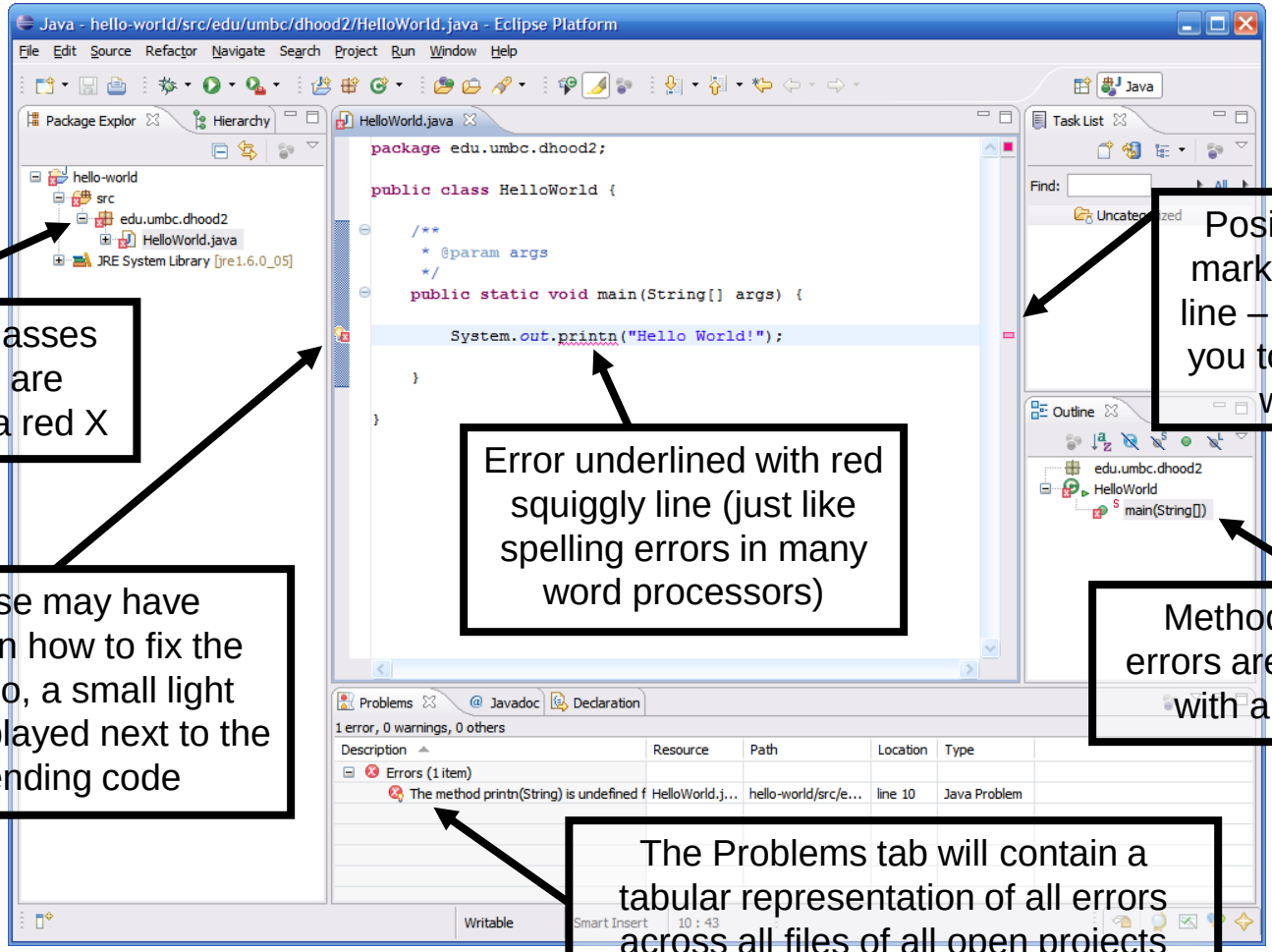


Compiling Source Code

- One huge feature of Eclipse is that it automatically compiles your code in the background
  - You no longer need to go to the command prompt and compile code directly
- This means that errors can be corrected when made
  - We all know that iterative development is the best approach to developing code, but going to shell to do a compile can interrupt the normal course of development
  - This prevents going to compile and being surprised with 100+ errors

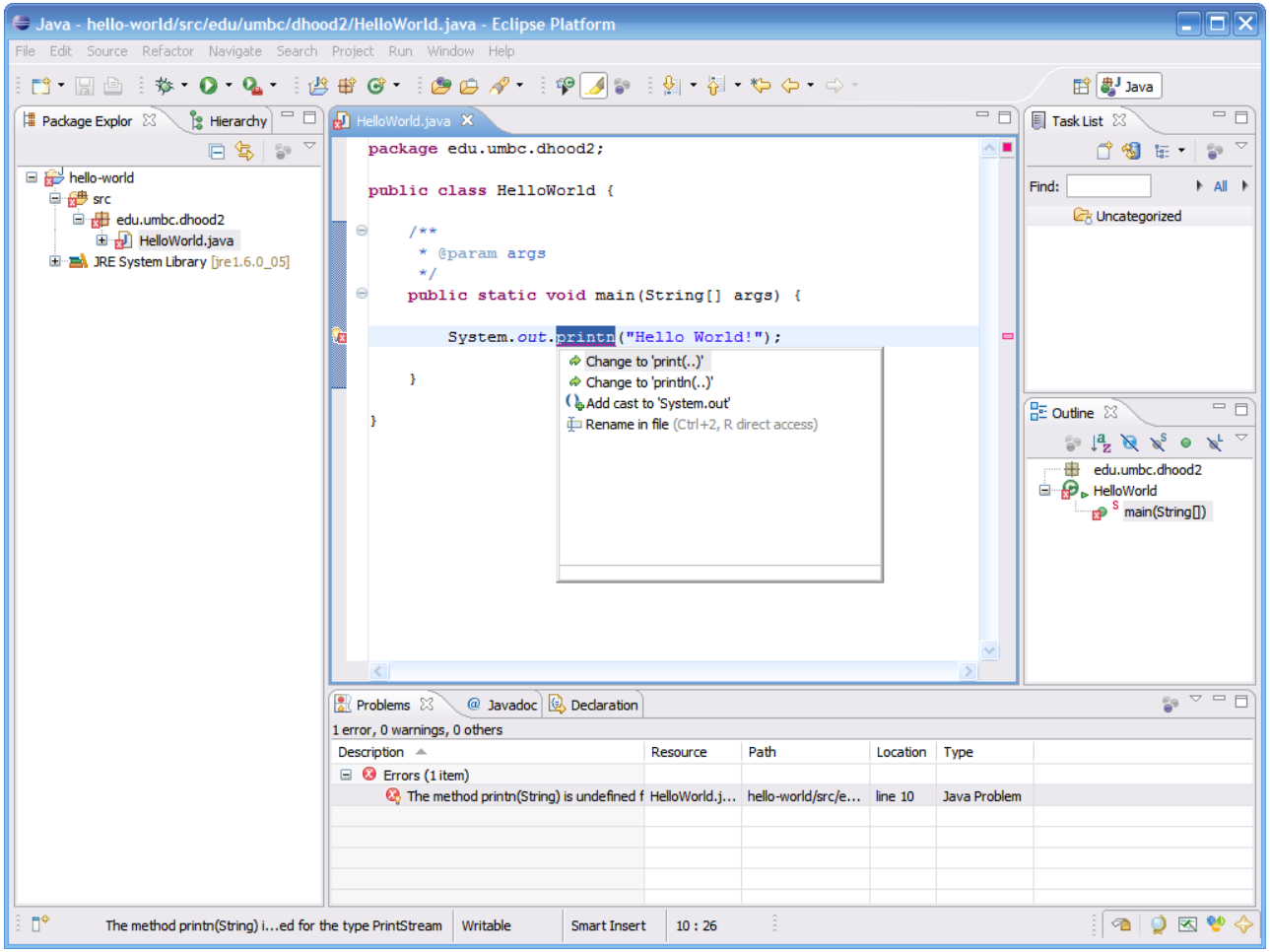
Example Compilation Error

- This code contains a typo in the println statement…



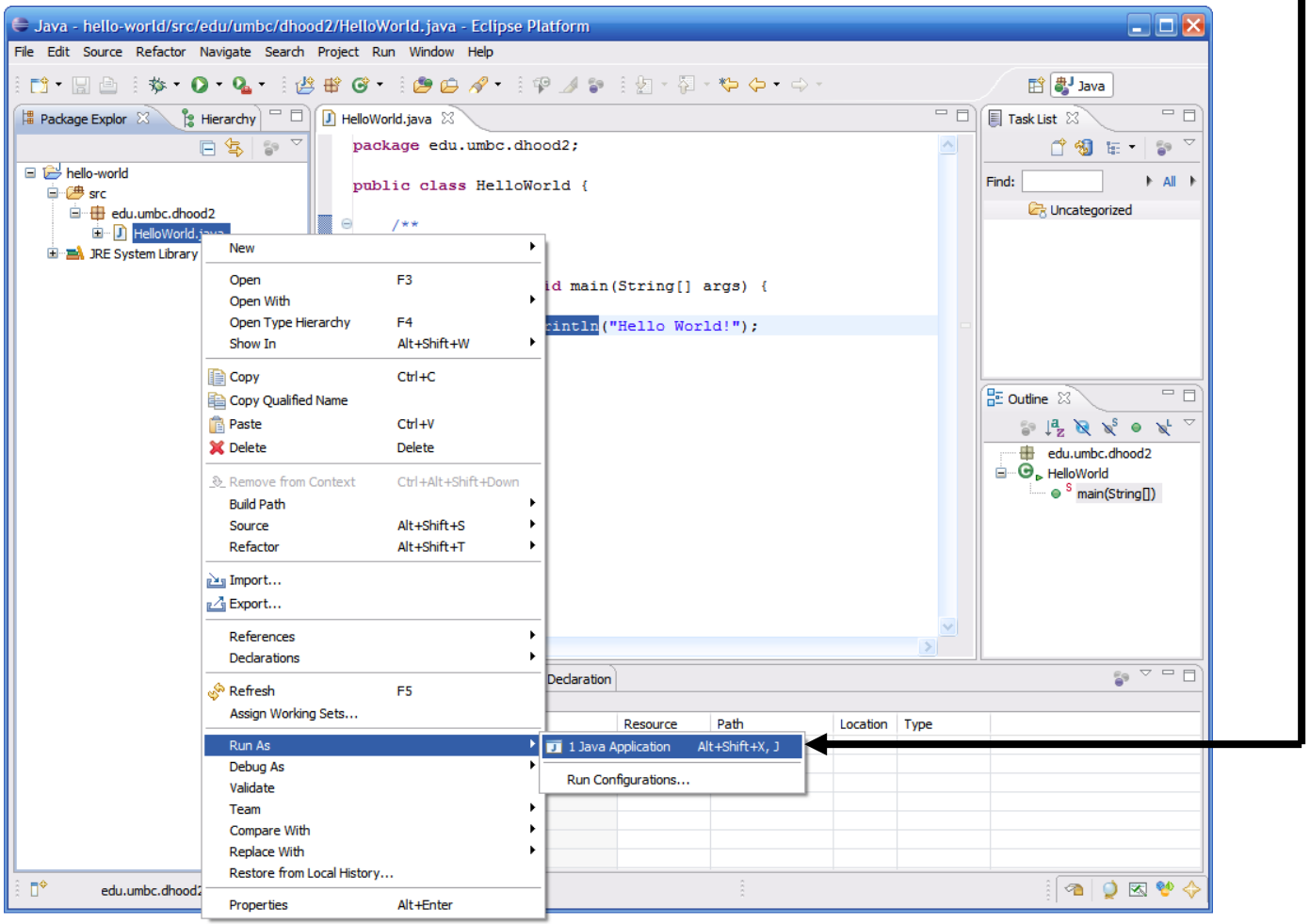
Example Compilation Error (continued)

- When clicking on the light bulb, Eclipse suggests changing printn to either print or println



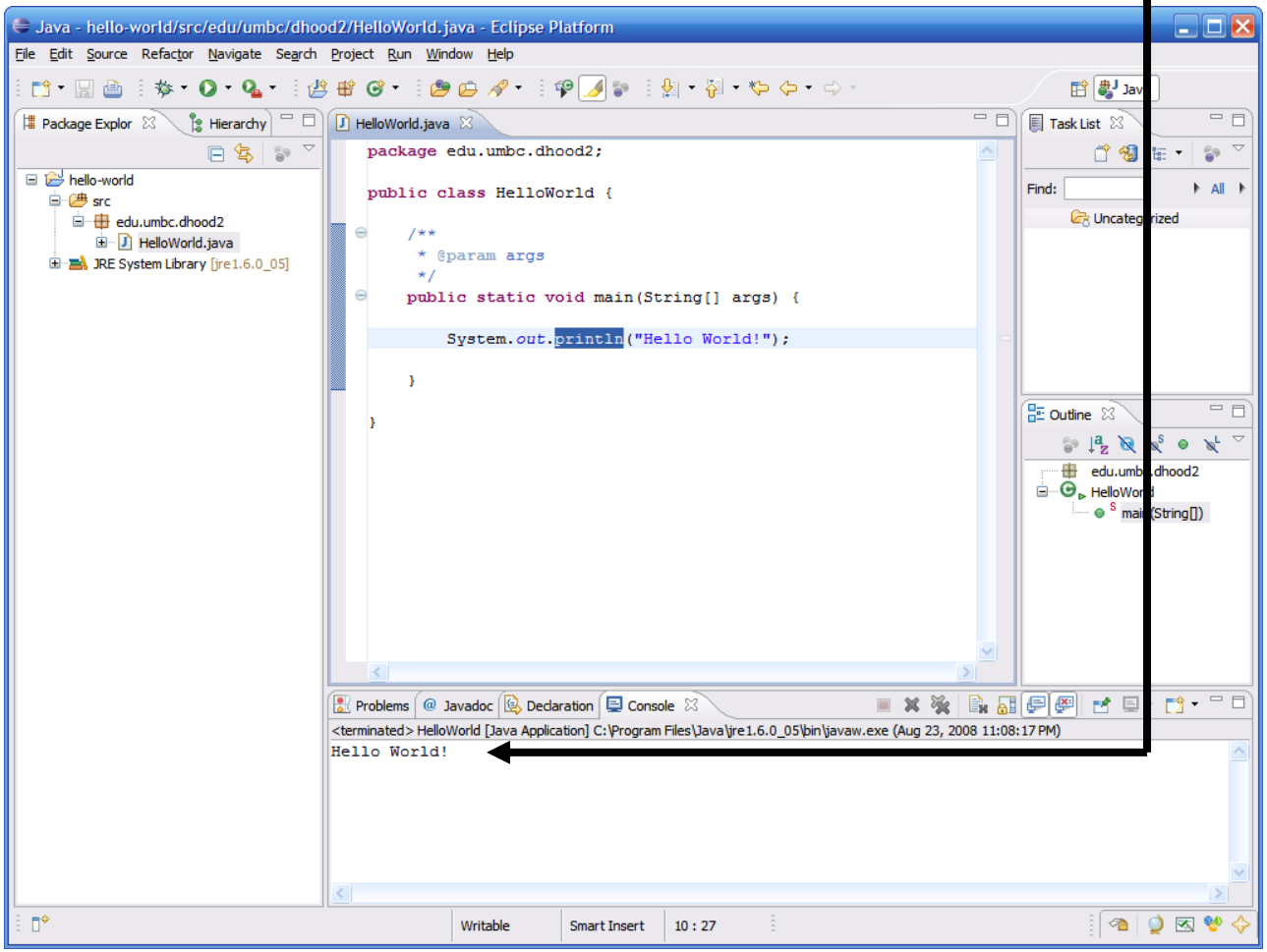
Running Code

- An easy way to run code is to right click on the class and select Run As → Java Application



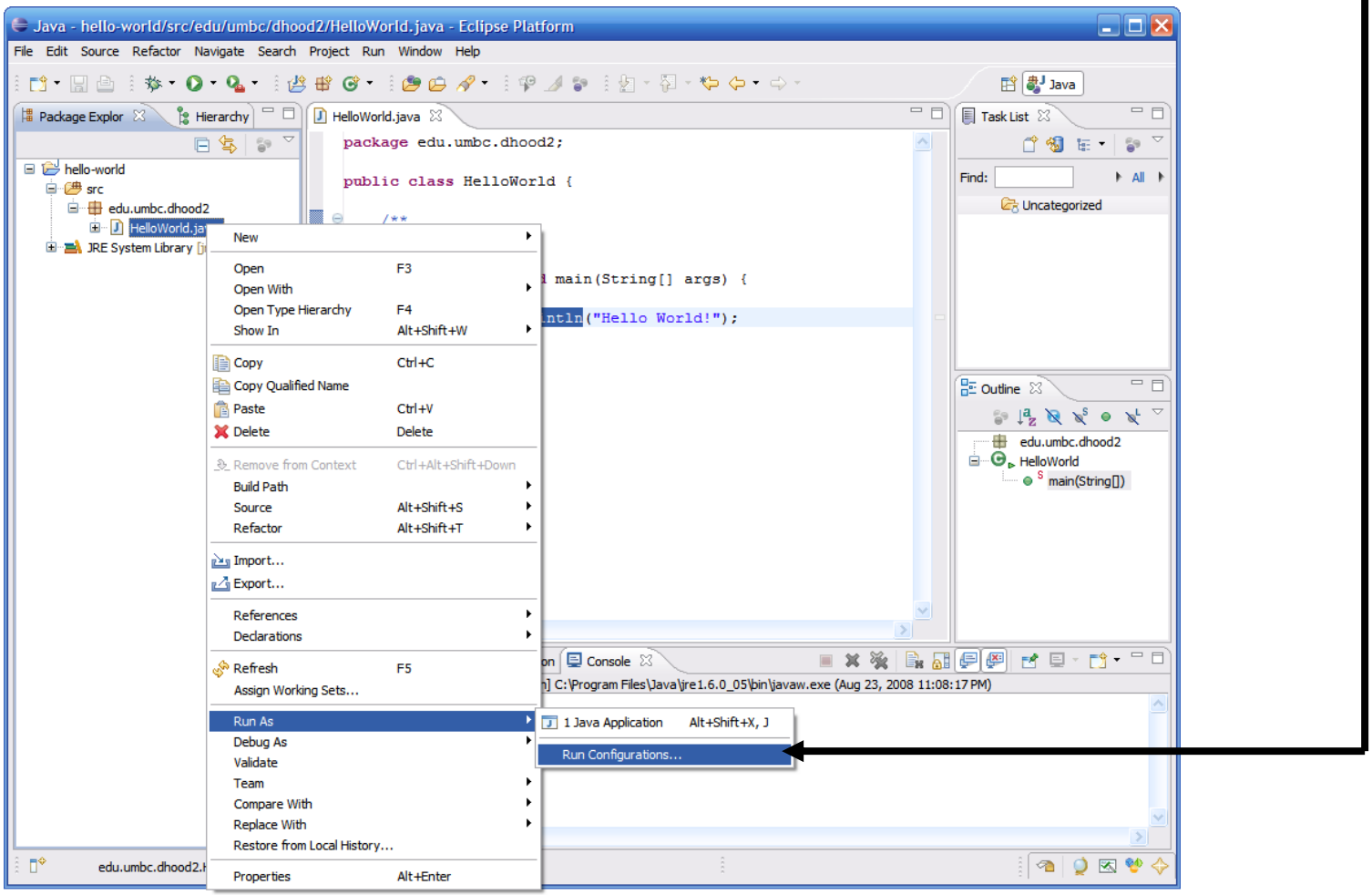
Running Code (continued)

- The output of running the code can be seen in the Console tab in the bottom pane

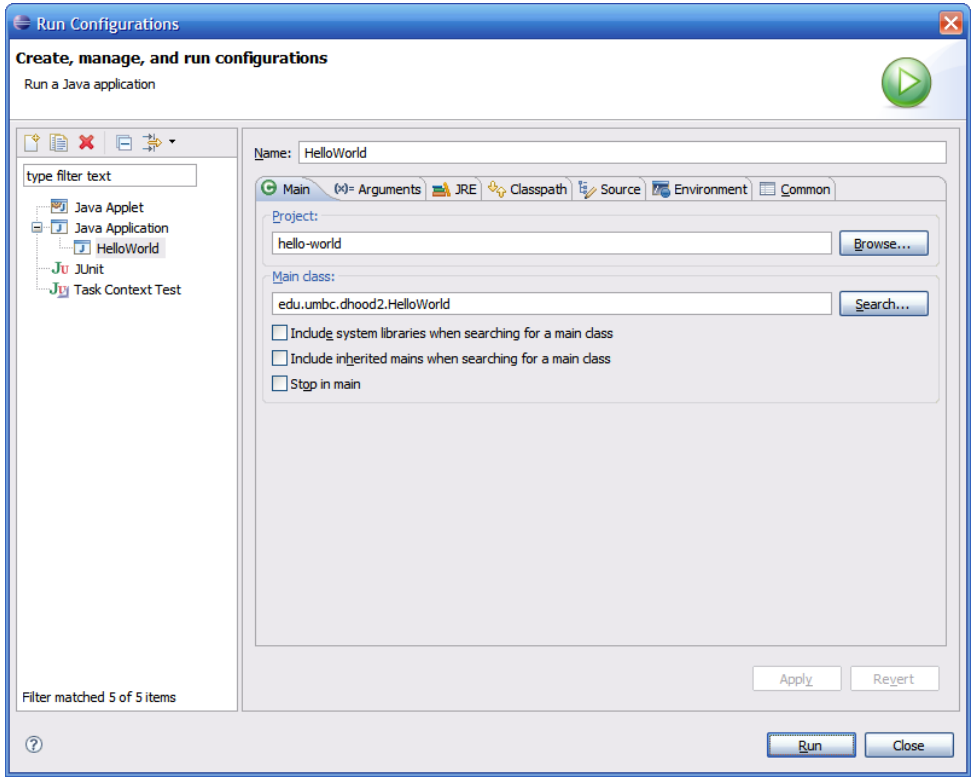


Run Configuration

- Advanced options for executing a program can be found by right clicking the class then clicking Run As → Run…



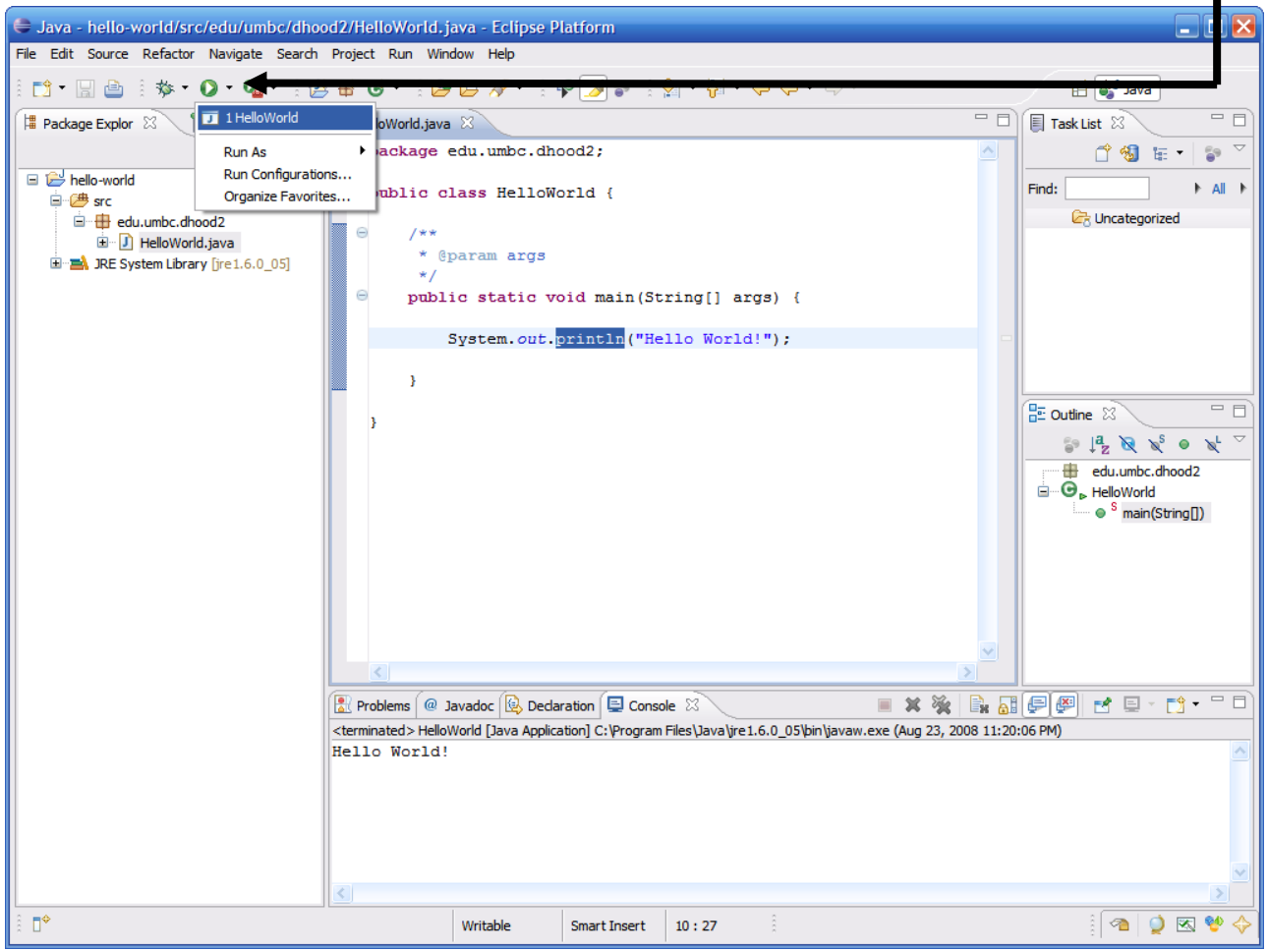
Run Configuration (continued)



- Here you can change/add any of the following:
  - JVM arguments
  - Command line arguments
  - Classpath settings
  - Environment variables
  - Which JVM to use

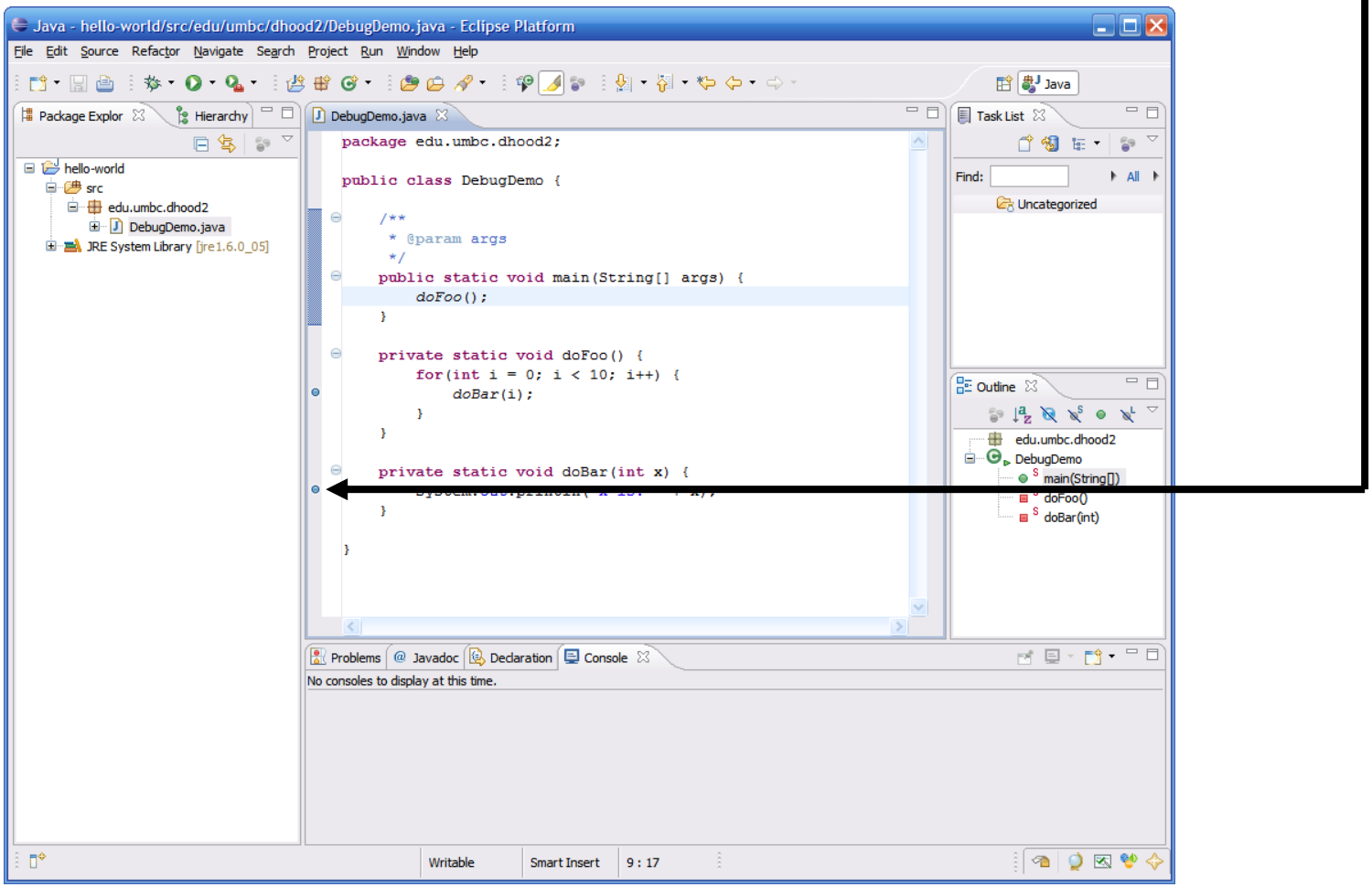
Re-Running Code

- After you run the code a first time, you can re-run it just by selecting it from the run drop down menu



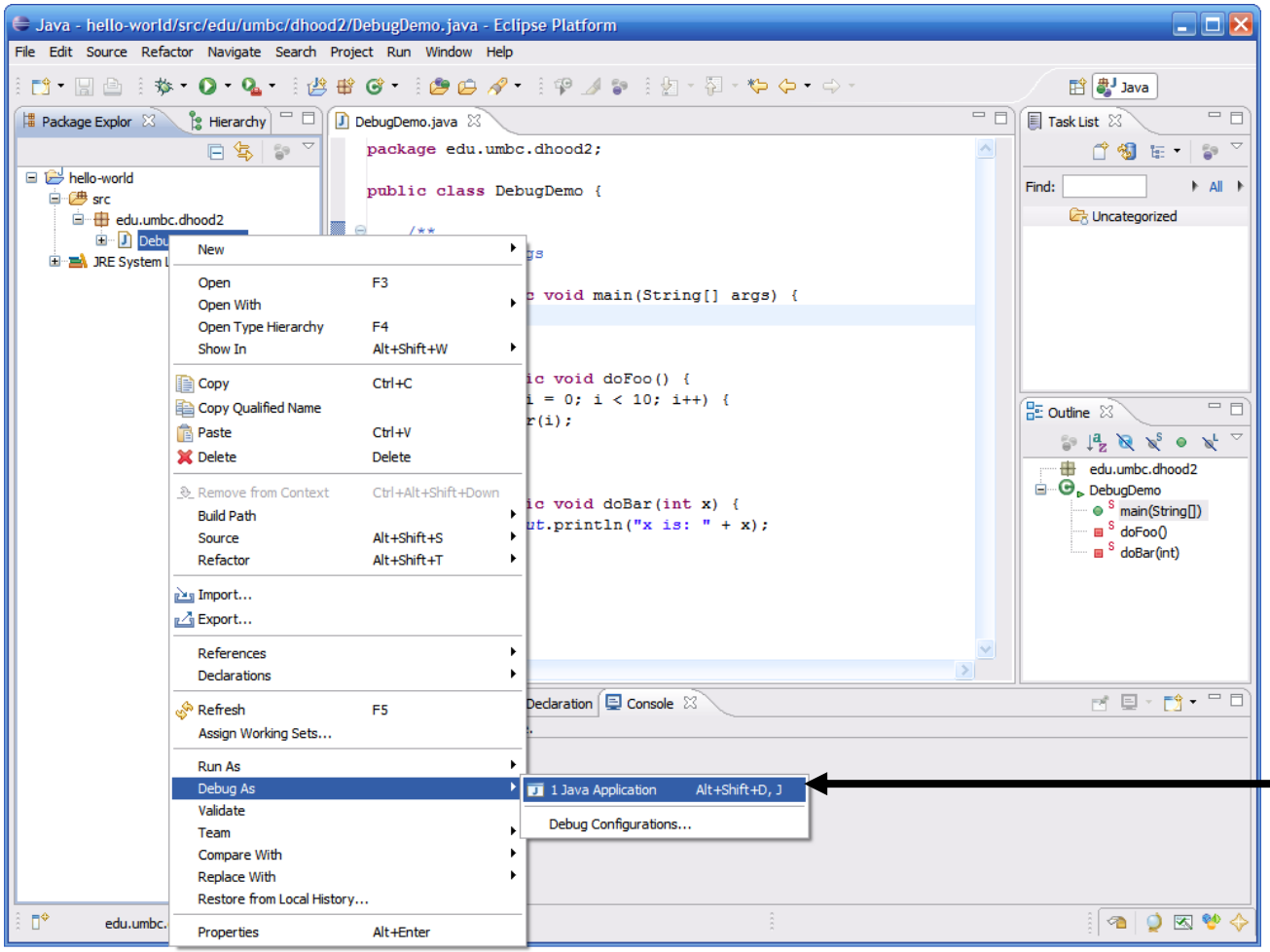
Debugging Code

- Eclipse comes with a pretty good built-in debugger
- You can set break points in your code by double clicking in the left hand margin – break points are represented by these blue bubbles



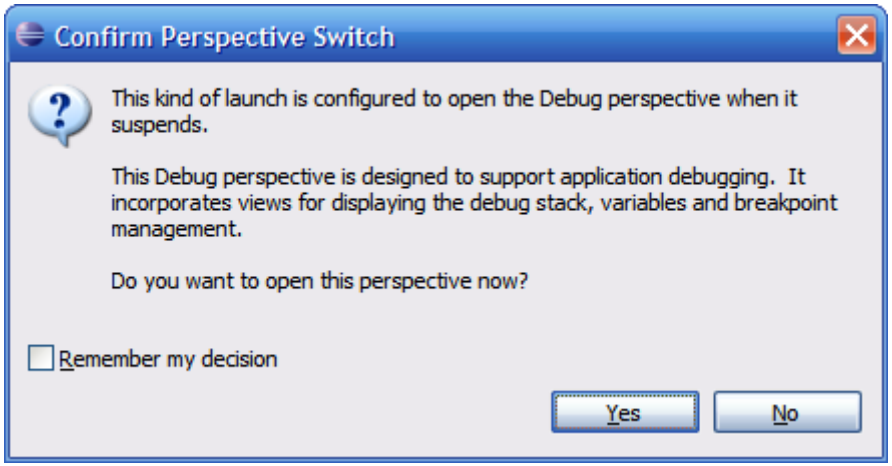
Debugging Code (continued)

- An easy way to enter debug mode is to right click on the class and select Debug As → Java Application



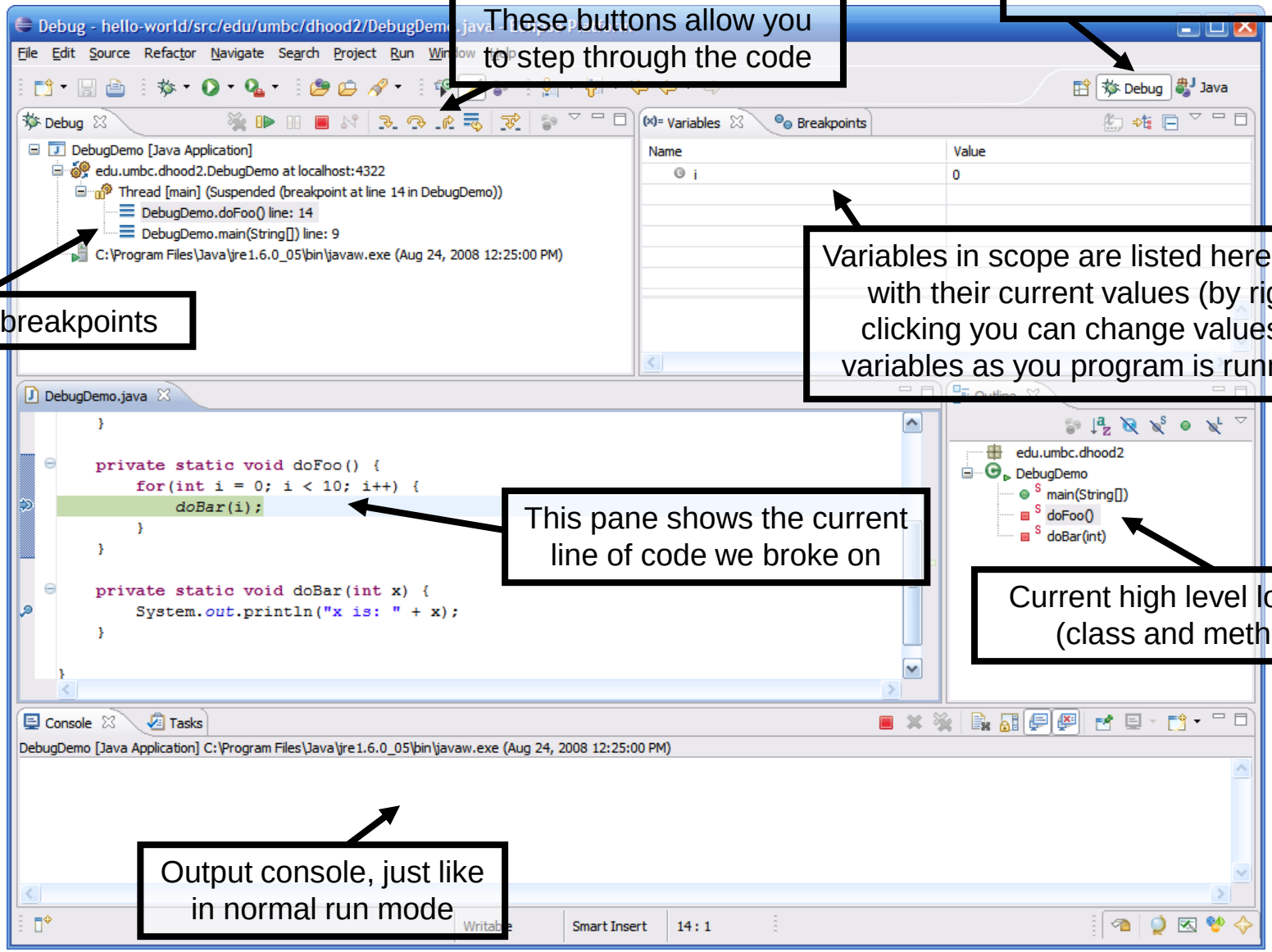
Debugging Code (Continued)

- The first time you try to debug code you will be presented with the following dialog



- Eclipse is asking if you want to switch to a perspective that is more suited for debugging, click Yes
- Eclipse has many perspectives based on what you are doing (by default we get the Java perspective)

Debug Perspective

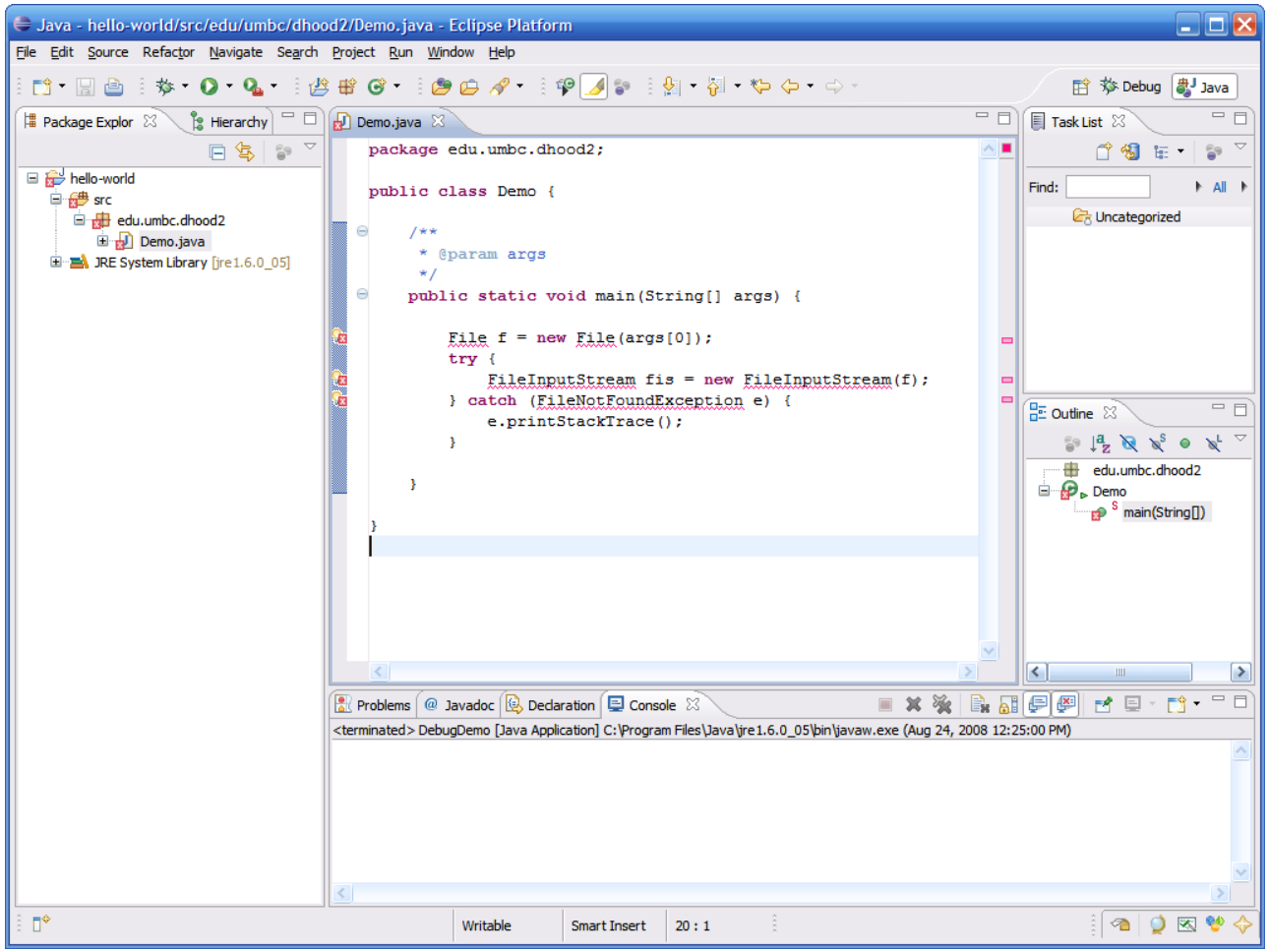


Sampling of Some Other Features

- Import organization
- Context assist
- Javadoc assist
- Getter/Setter generation
- Add unimplemented methods
- Exception handling
- Reminders
- Local history

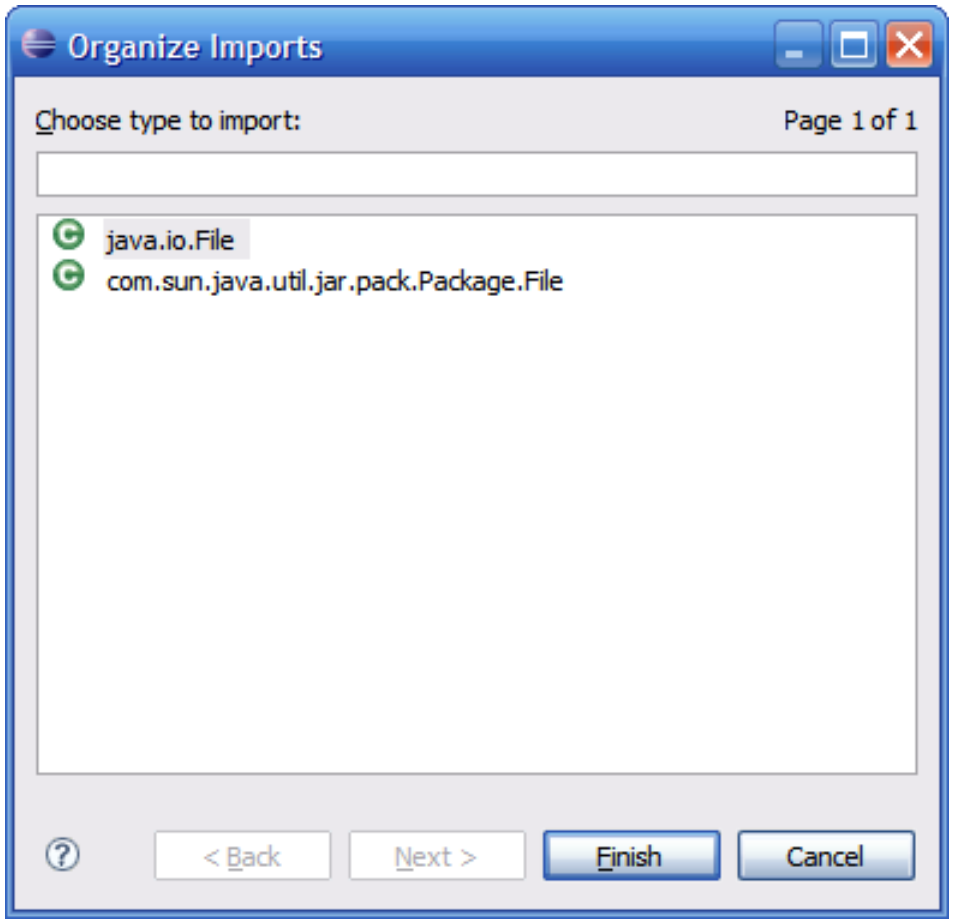
Import Organization

- Eclipse can automatically include import statements for any classes you are using, just press Control + Shift + o (letter o)



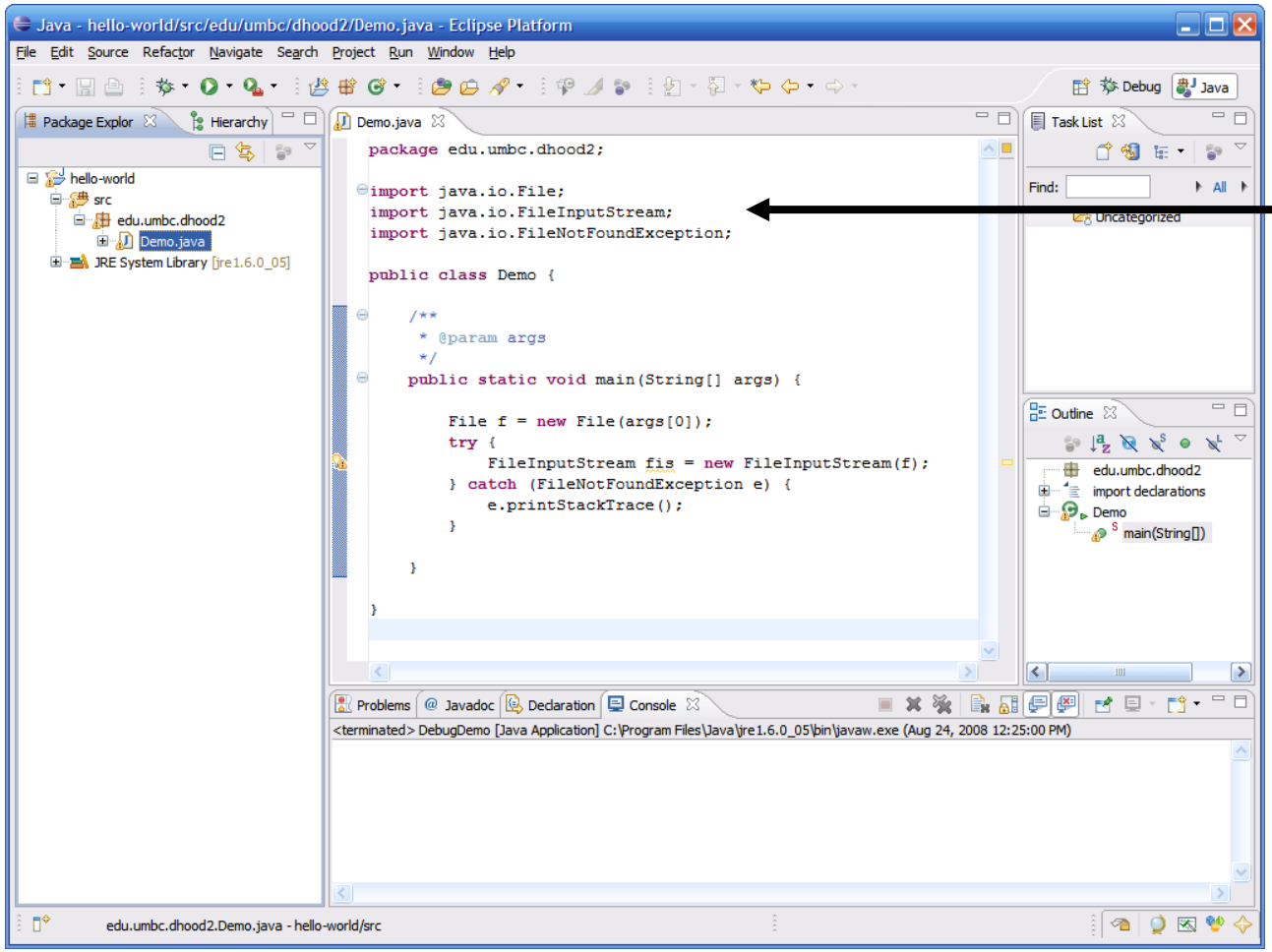
Import Organization (continued)

- If the class is ambiguous (more than one in the API) then it will ask you to select the correct one



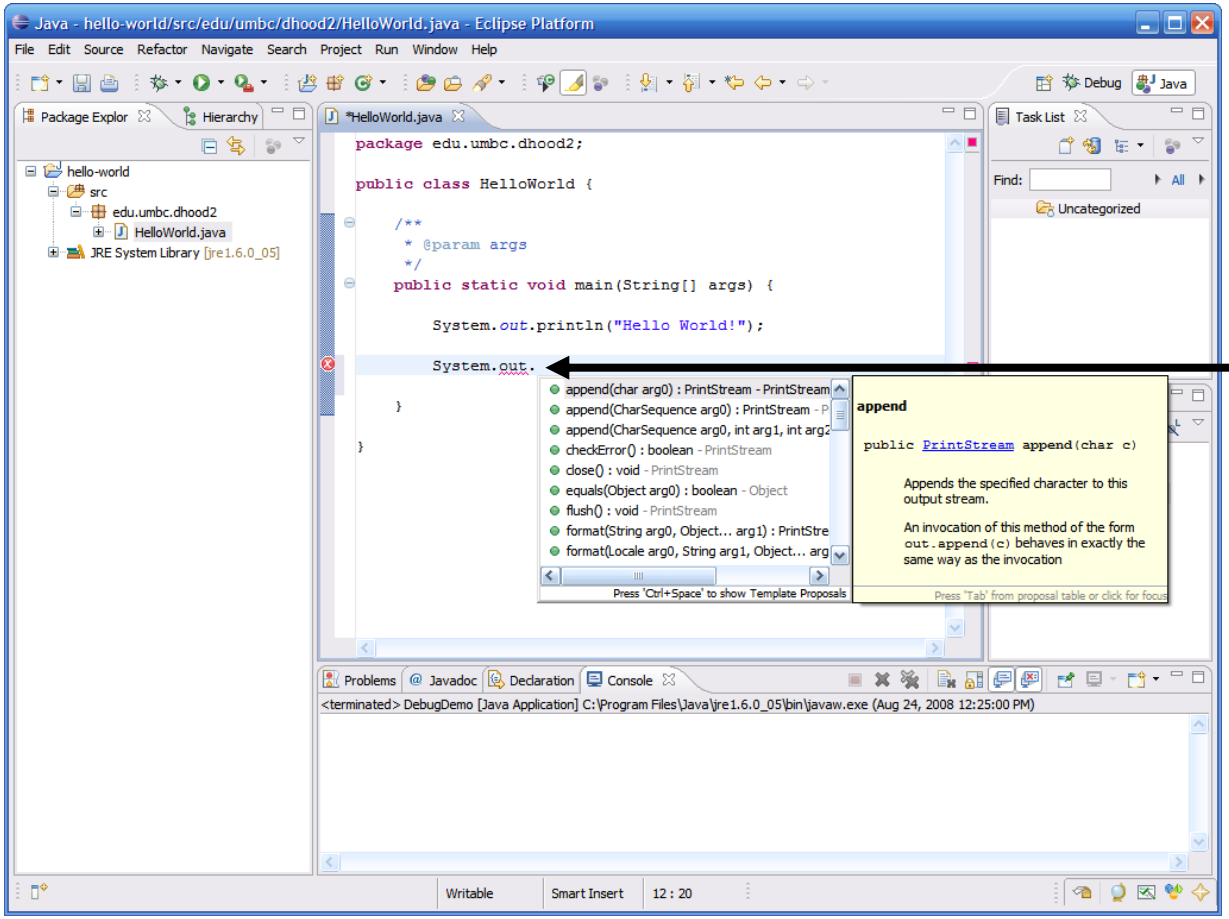
Import Organization (continued)

- Import statements automatically included and organized
  - You can organize imports to clean them up at any time



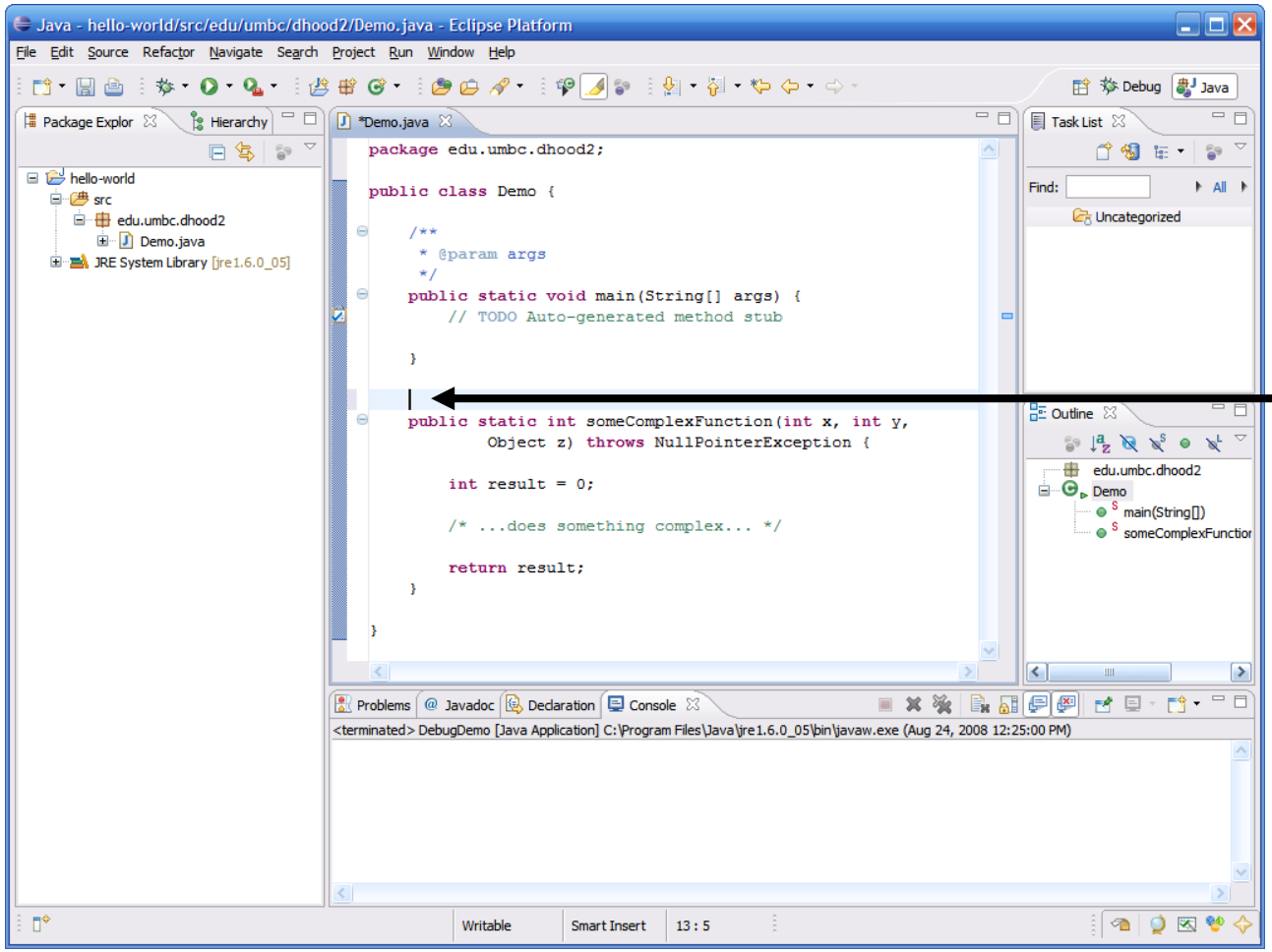
Context Assist

- If you are typing and press a "." character and pause a second, Eclipse will show you a list of all available methods for the class
  - Prevents having to browse javadocs to see what methods are available
  - Get context assist at any time by pressing Control + Space



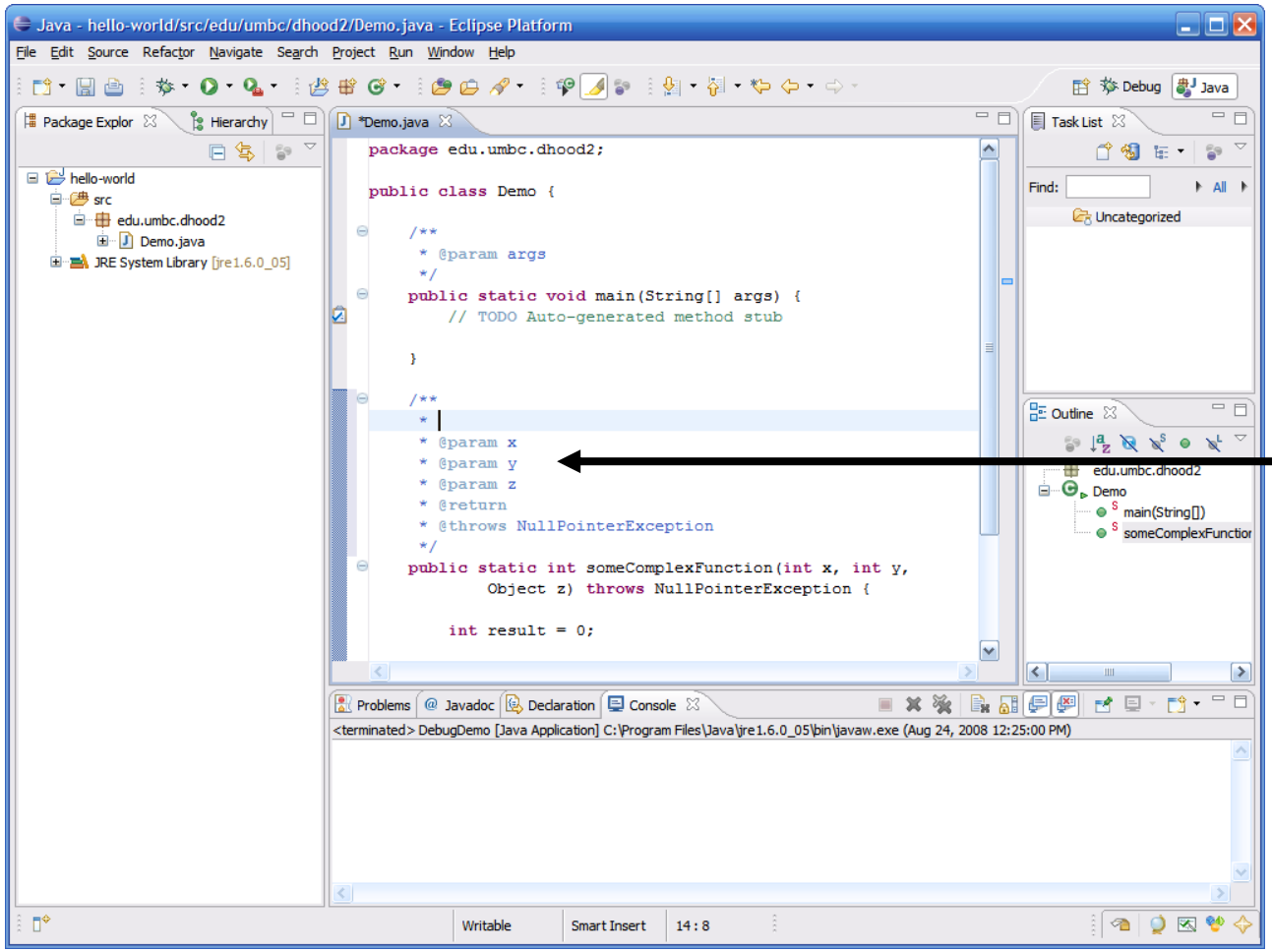
Javadoc Assist

- Eclipse can also help generate javadoc comments for you, simply place the cursor before the method and then type “/**” then Enter



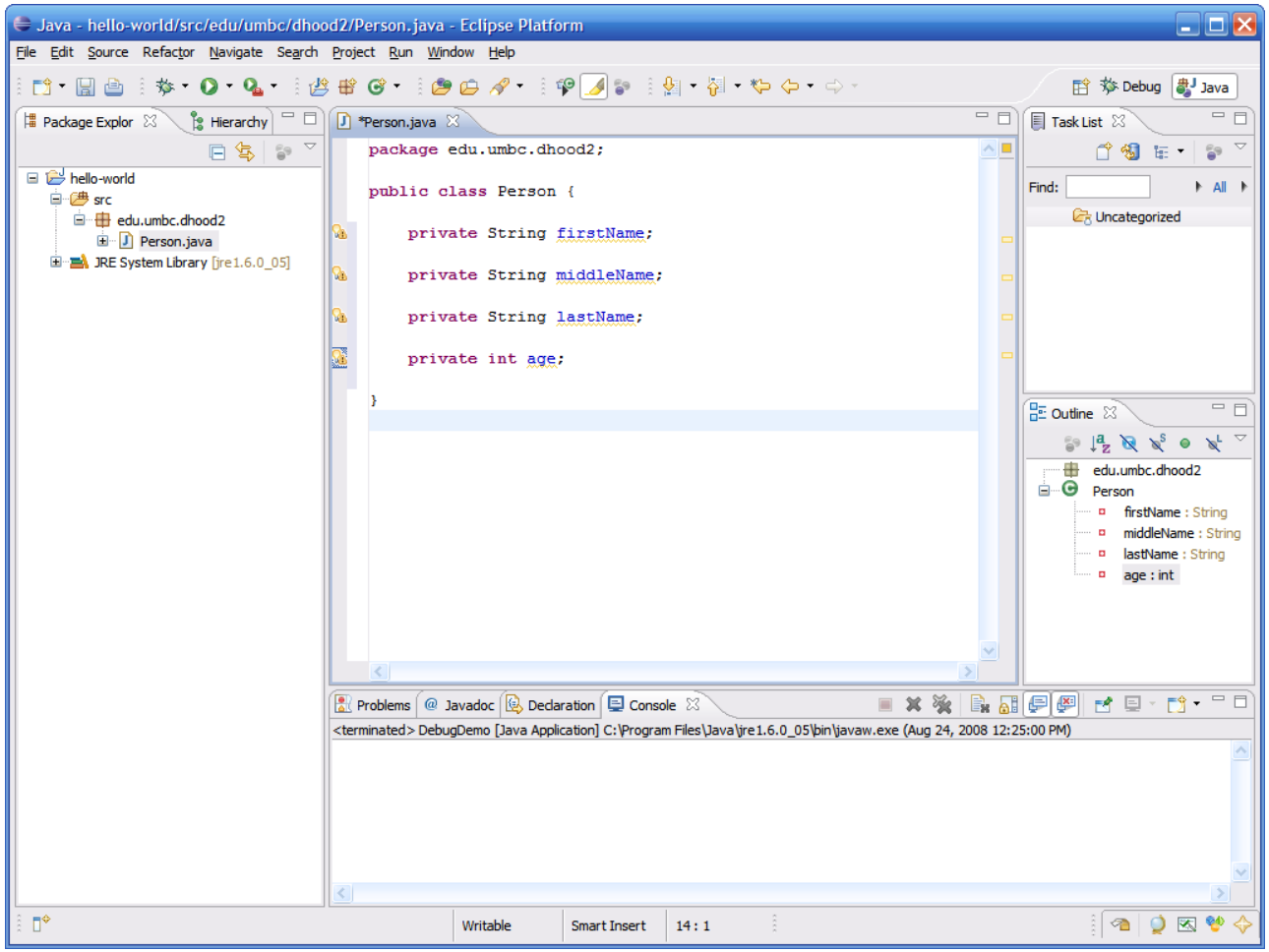
Javadoc Assist (continued)

- Eclipse will automatically generate a javadoc header for the method all stubbed out with the parameters, return type and exceptions



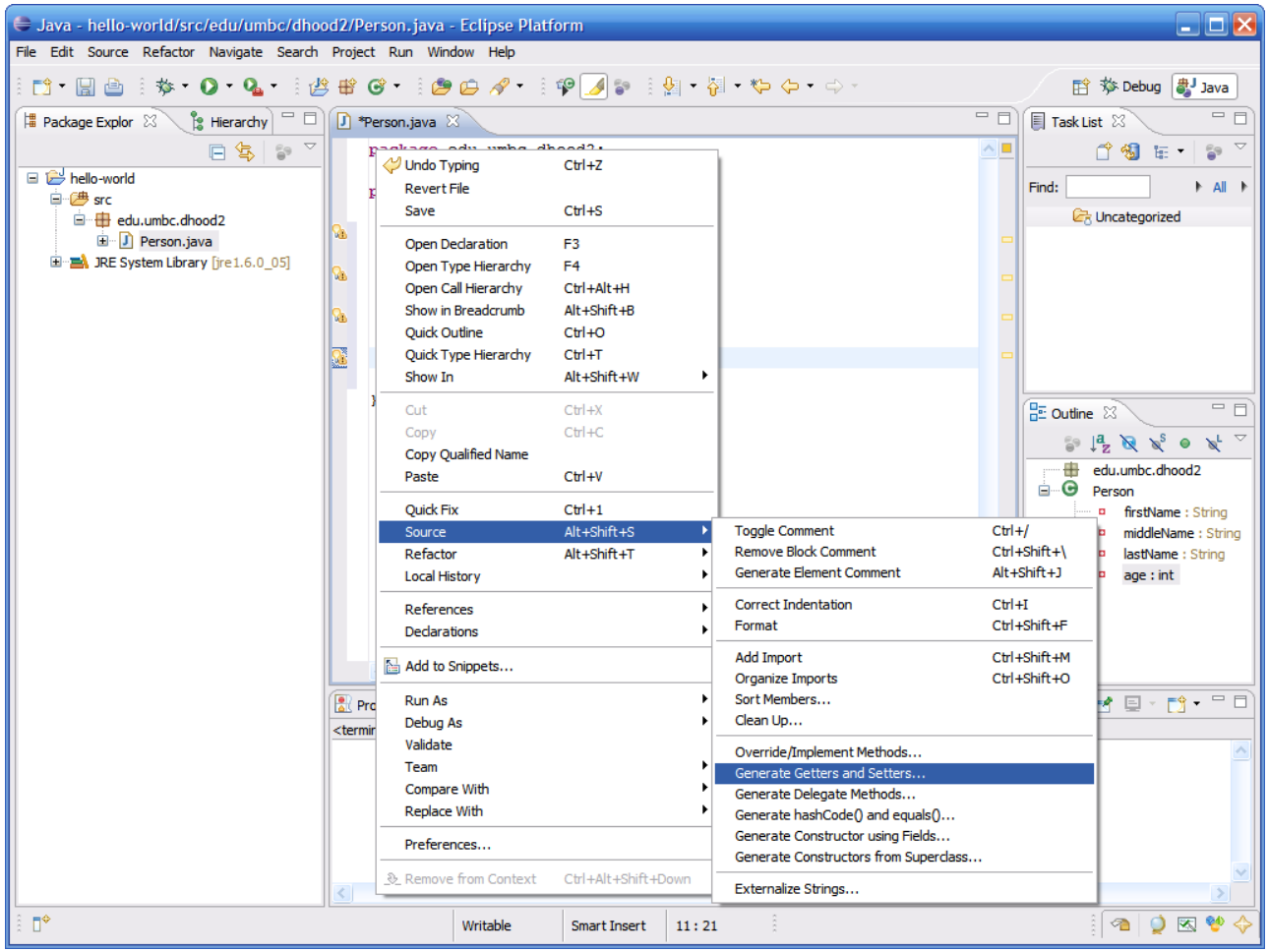
Getter/Setter Generation

- Eclipse can automatically generate getters and setters for member of a class…



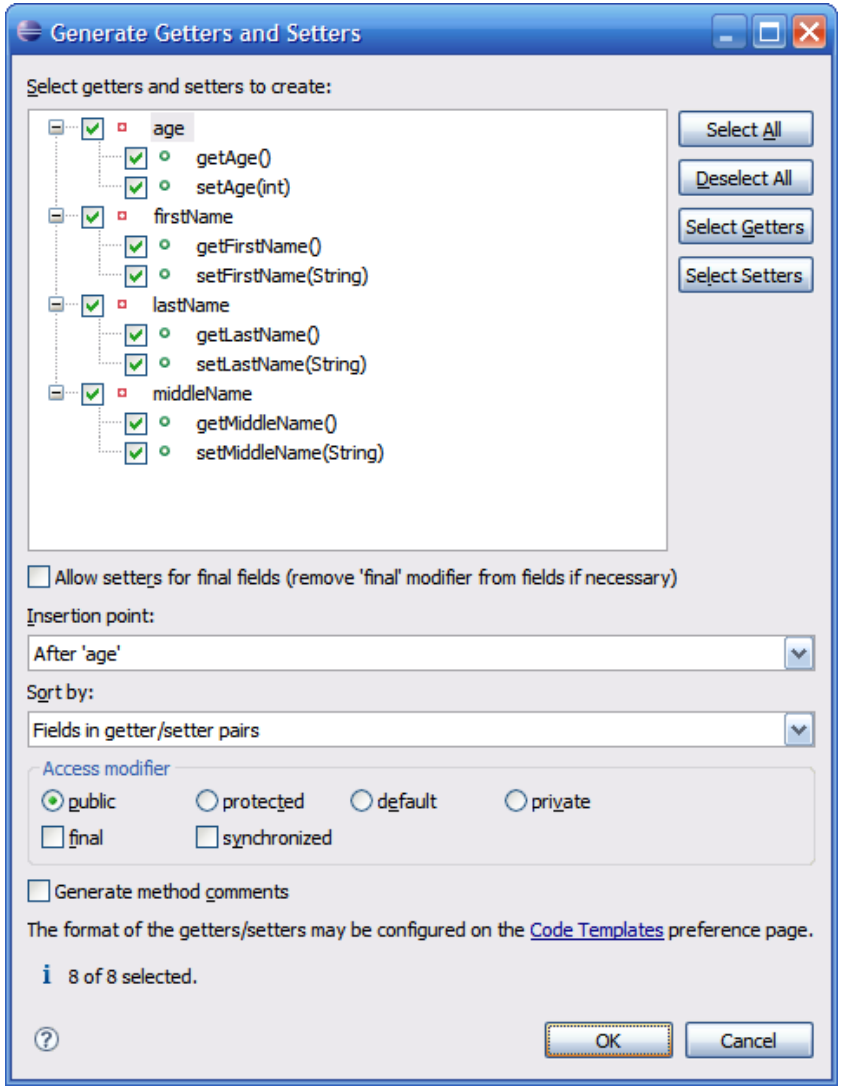
Getter/Setter Generation (continued)

- To generate getters and setters, right click in the main pane, then select Source → Generate Getters and Setters



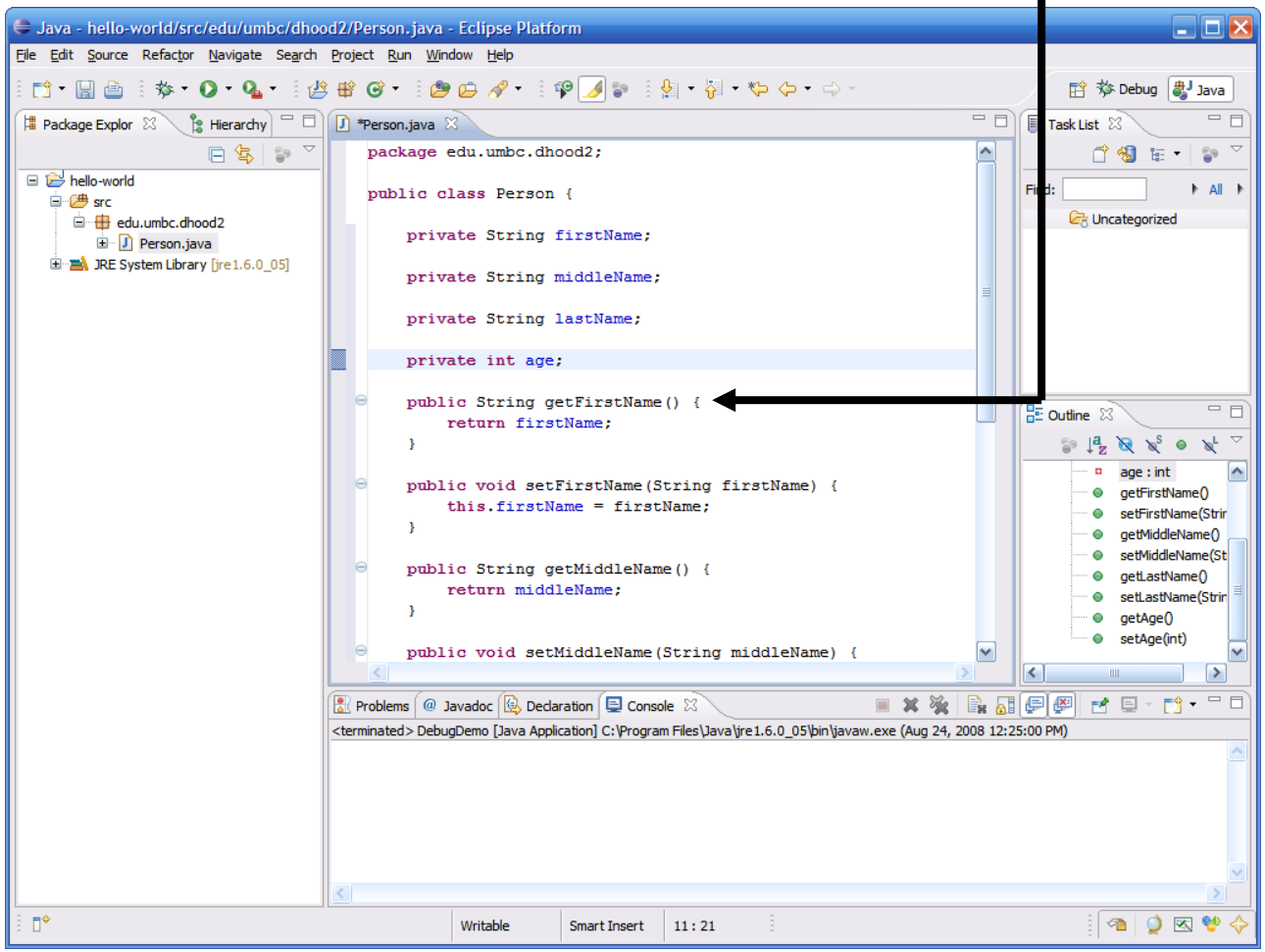
Getter/Setter Generation (continued)

- Here you can selectively choose members for which to generate getters and setters



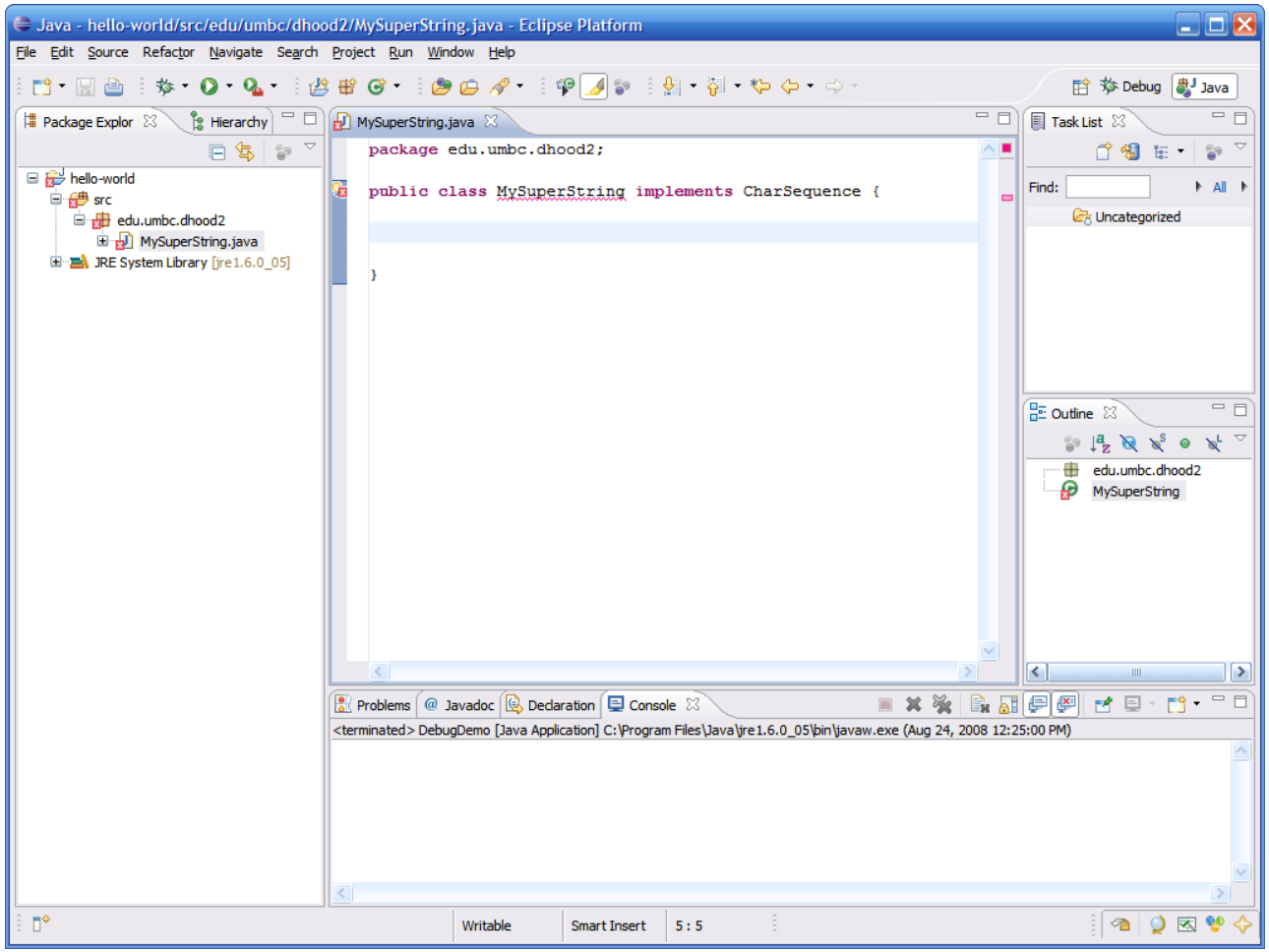
Getter/Setter Generation (continued)

- Eclipse will then automatically generate the code for the getters and setters



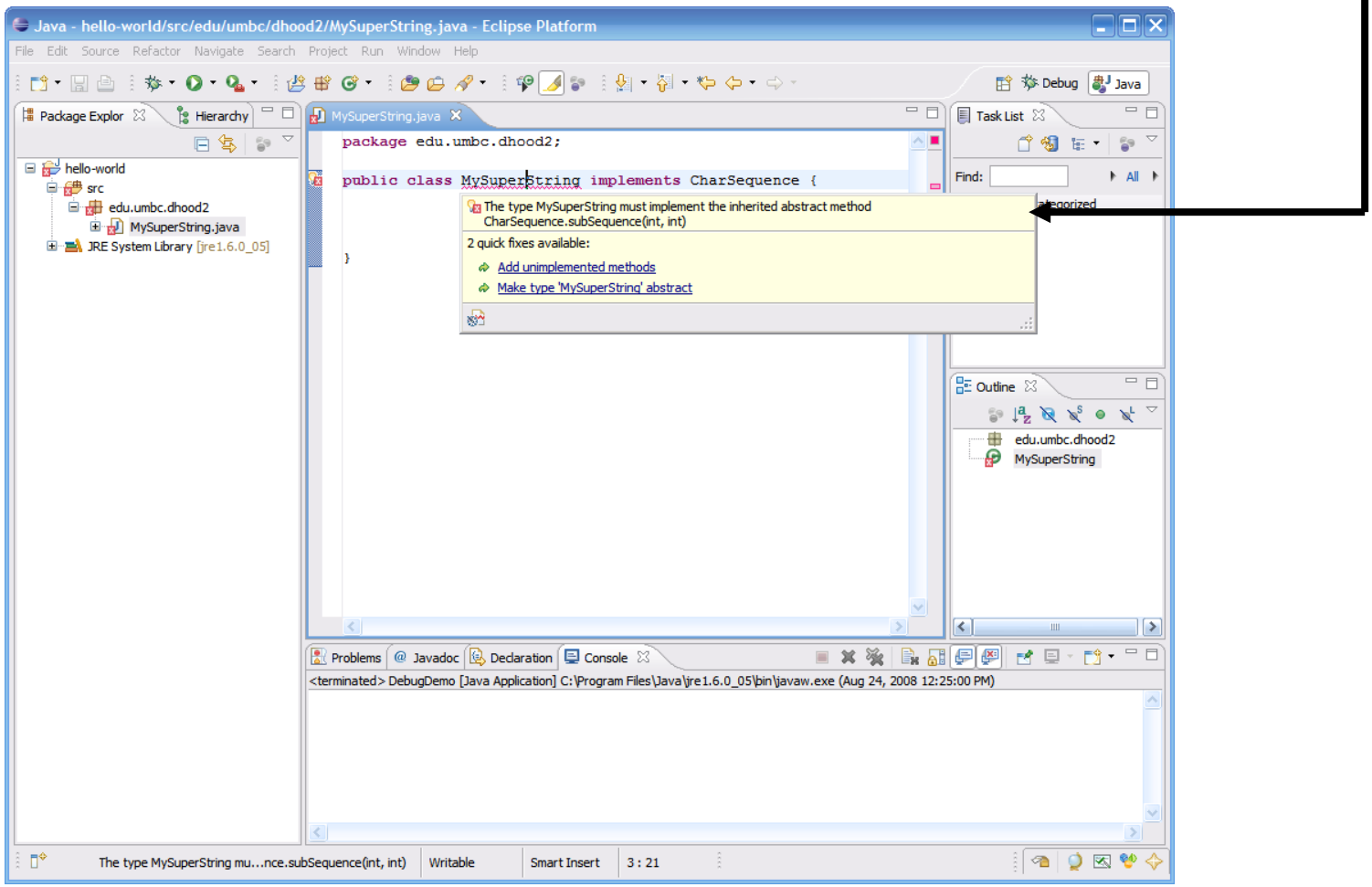
Add Unimplemented Methods

- Eclipse can also stub out methods that need to be present as a result of implementing an interface…



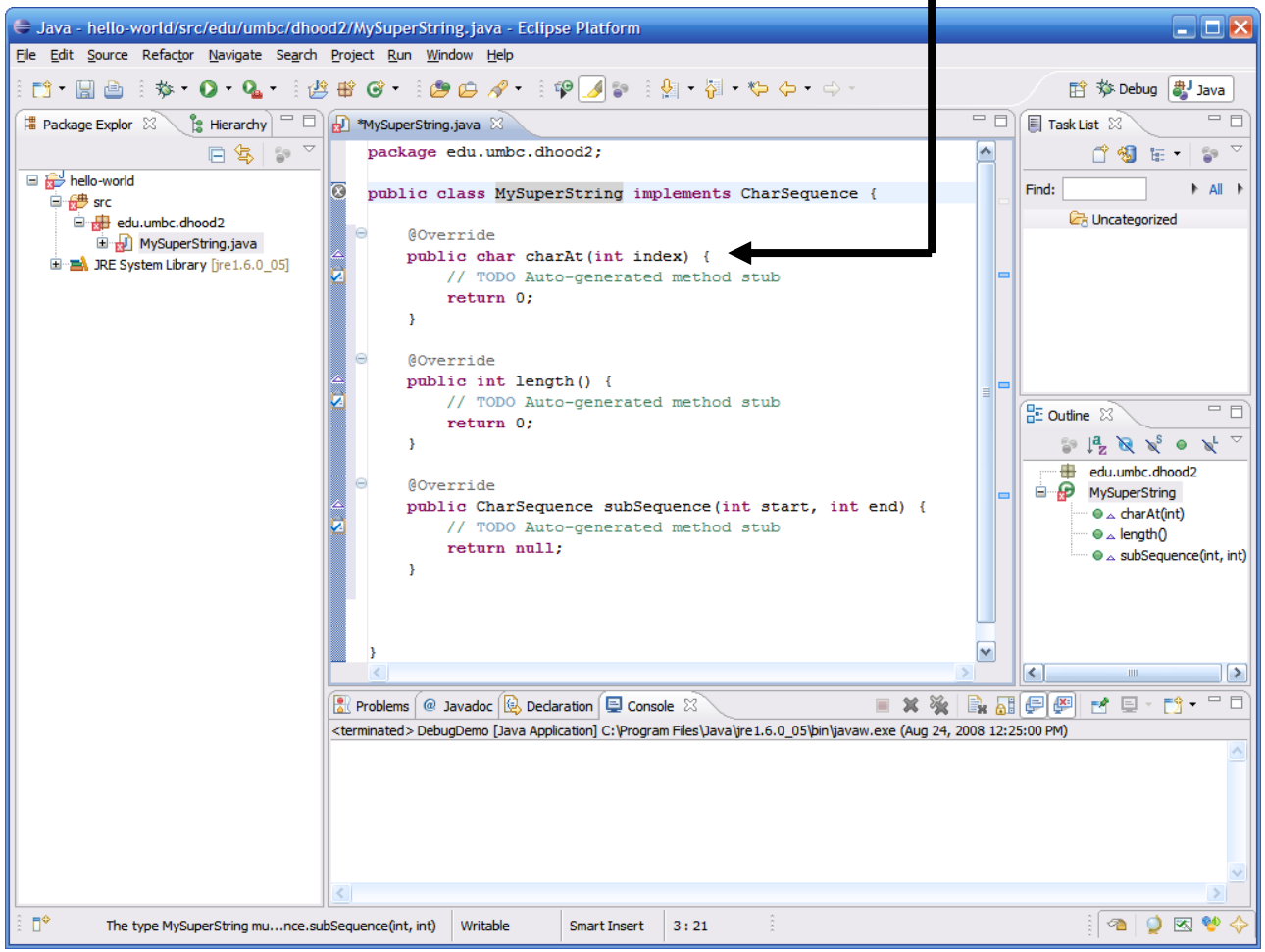
Add Unimplemented Methods (continued)

- You can use the quick fix light bulb to add the interfaces unimplemented methods to the class



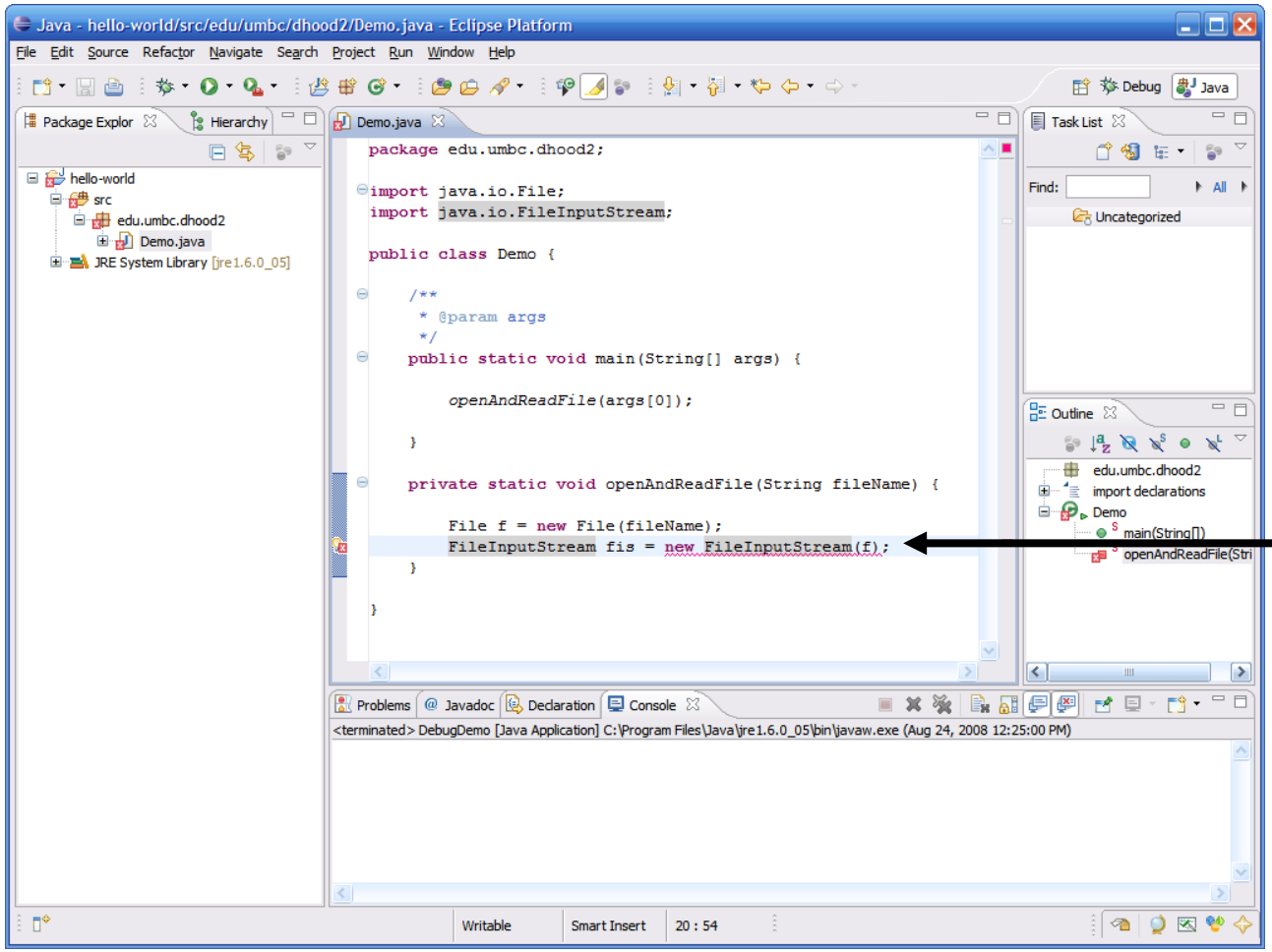
Add Unimplemented Methods (continued)

- Again Eclipse will go ahead and stub out the method for us



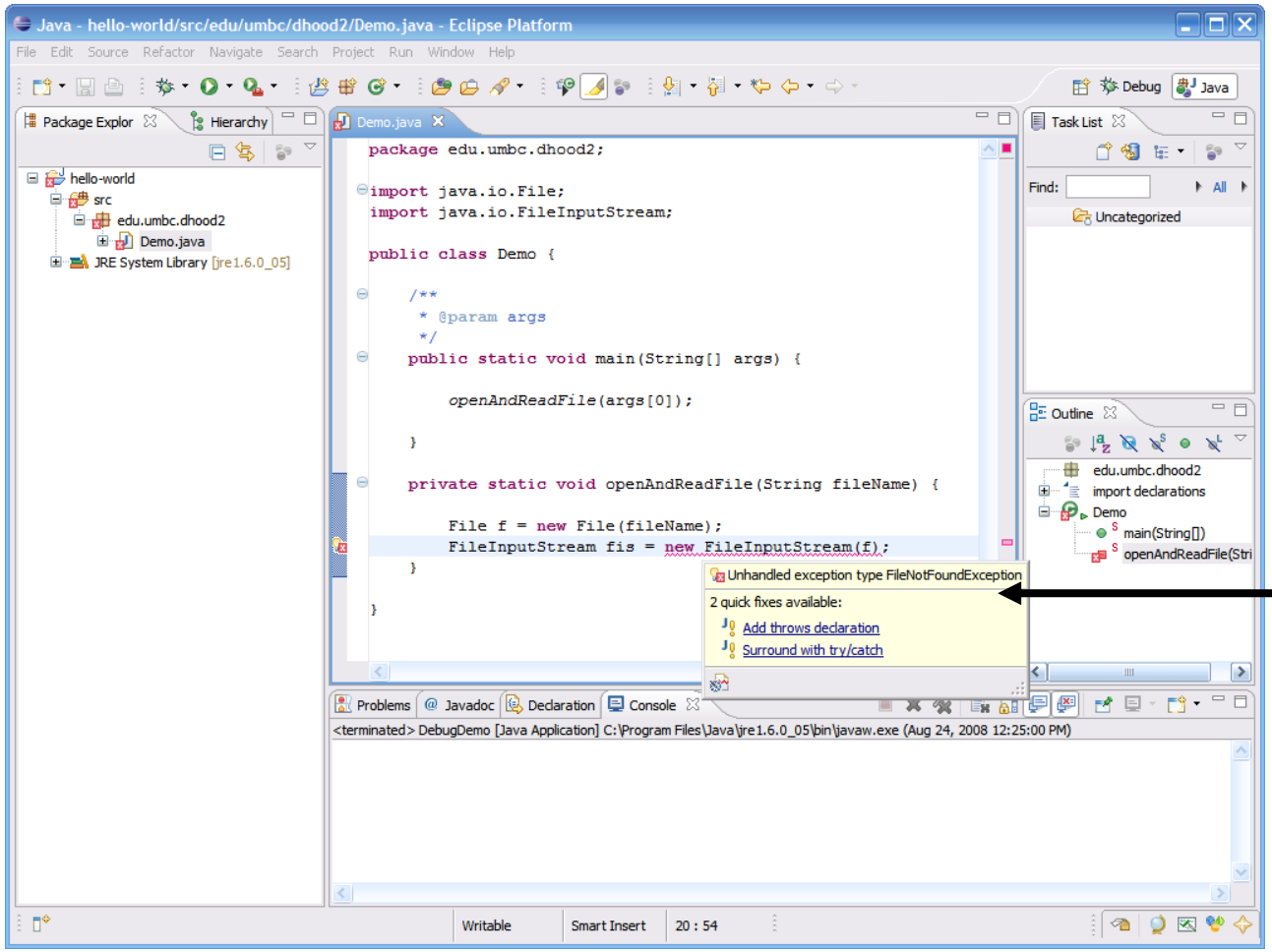
Exception Handling

- Eclipse will also pickup on unhandled exceptions



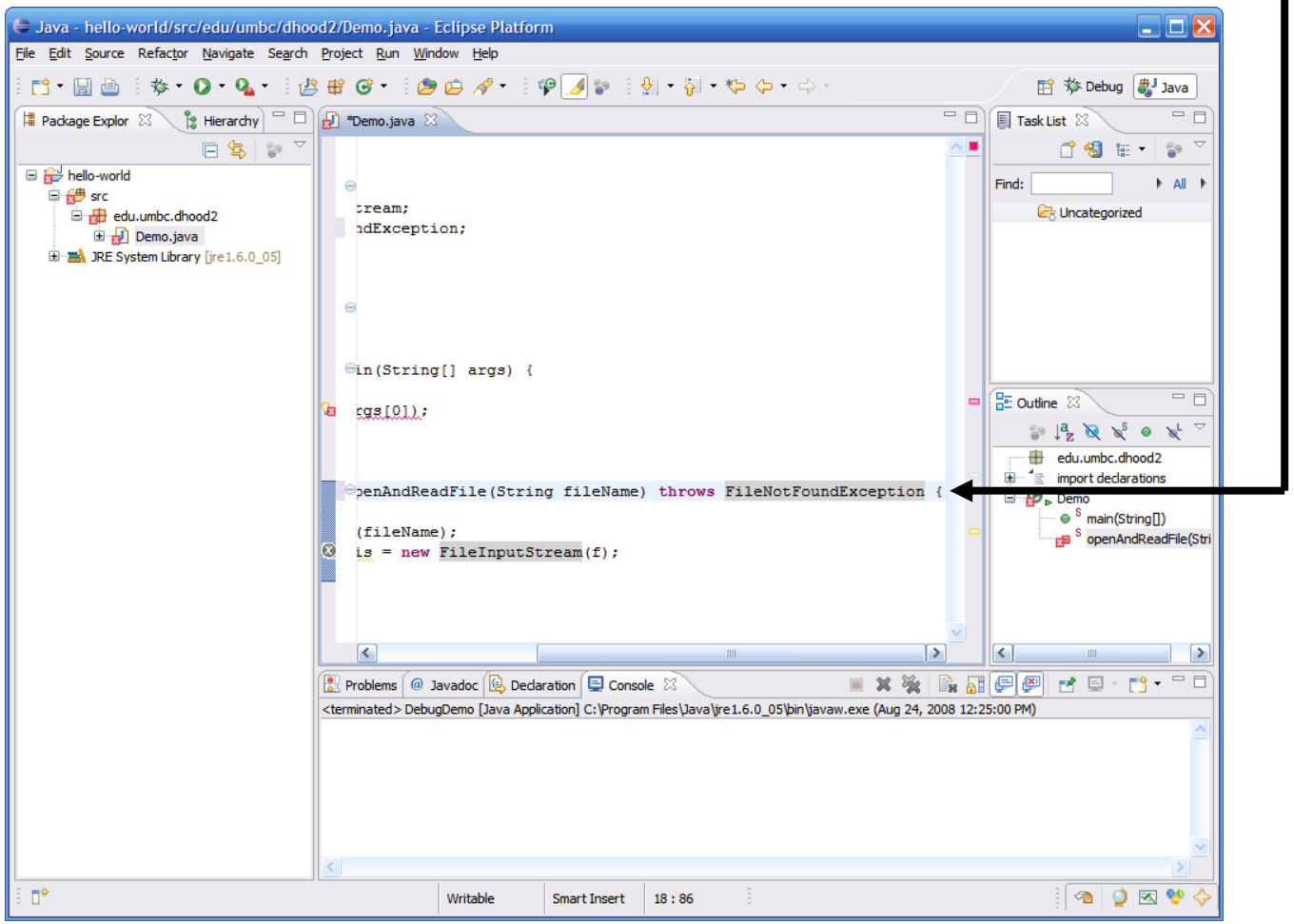
Exception Handling (continued)

- By clicking on the quick fix light bulb, Eclipse can suggest what to do to handle the exception



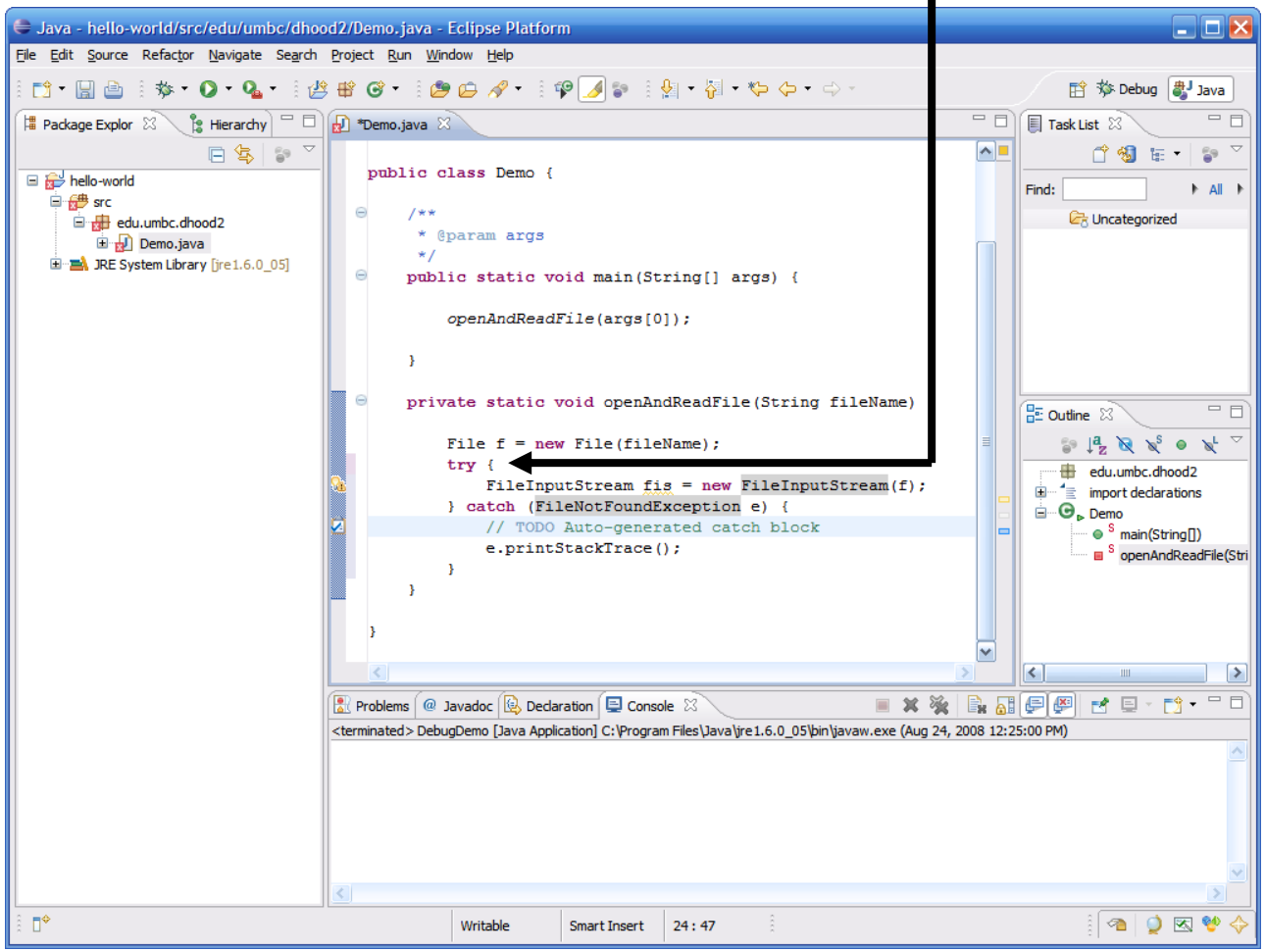
Exception Handling (continued)

- Eclipse can automatically add a “throws declaration” to the method signature



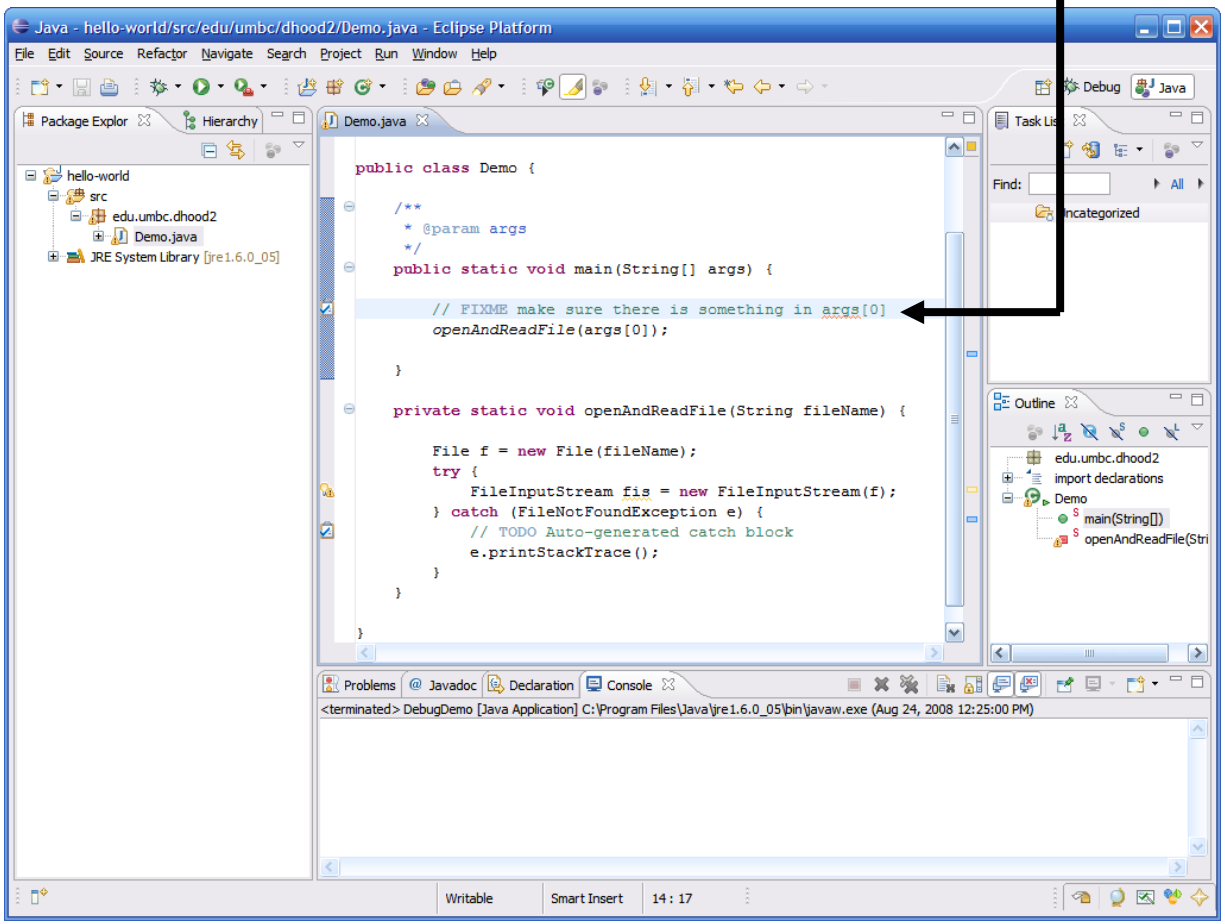
Exception Handling (continued)

- Alternately, Eclipse can also wrap the code inside a try/catch block



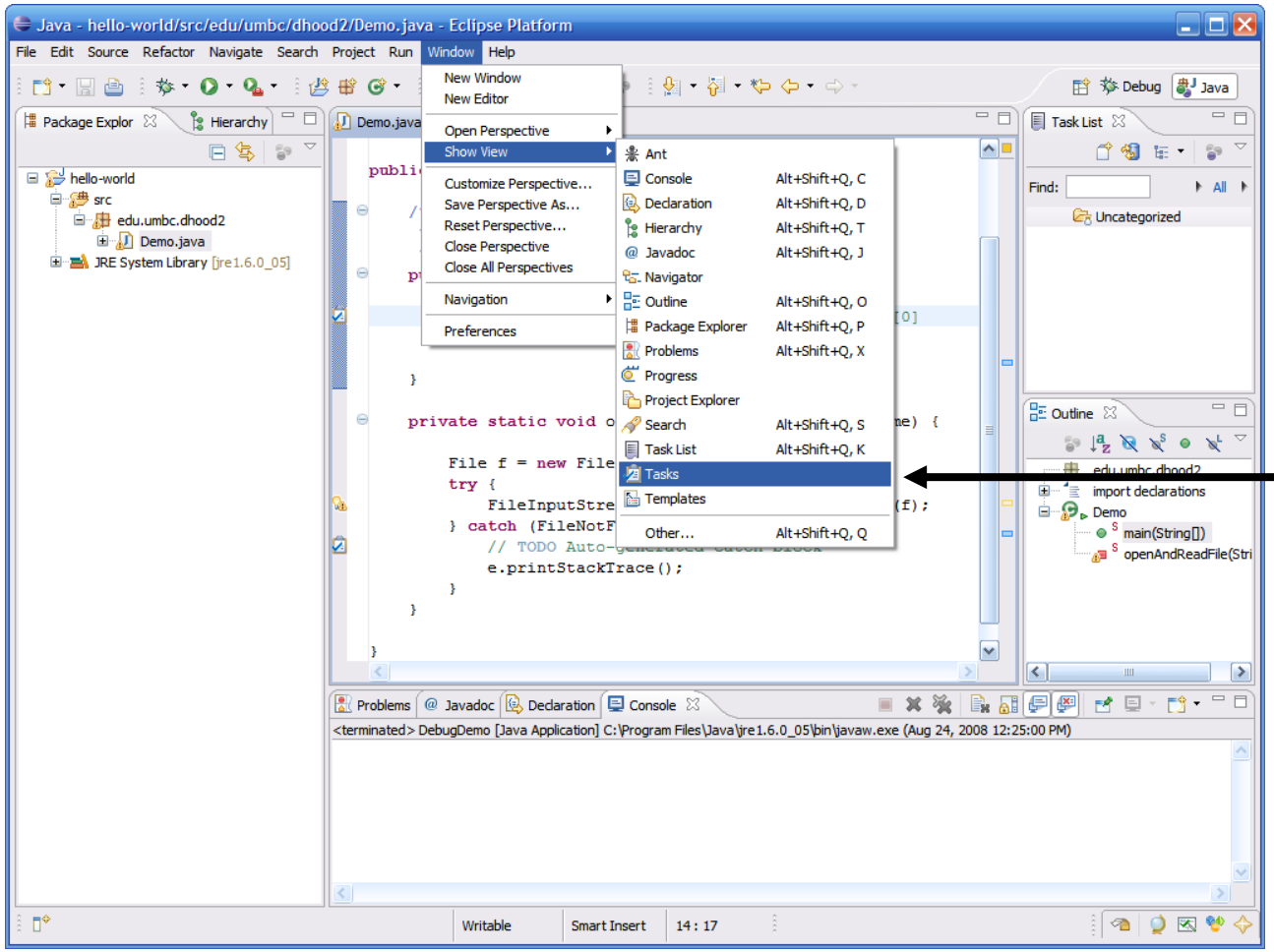
Tasks

- Eclipse allows you to insert reminders into your code and stores them for you to come back and revisit them
- Eclipse recognizes the following tags inside comments…
  - TODO
  - FIXME
  - XXX
- You can even add your own custom tasks through the preferences menu



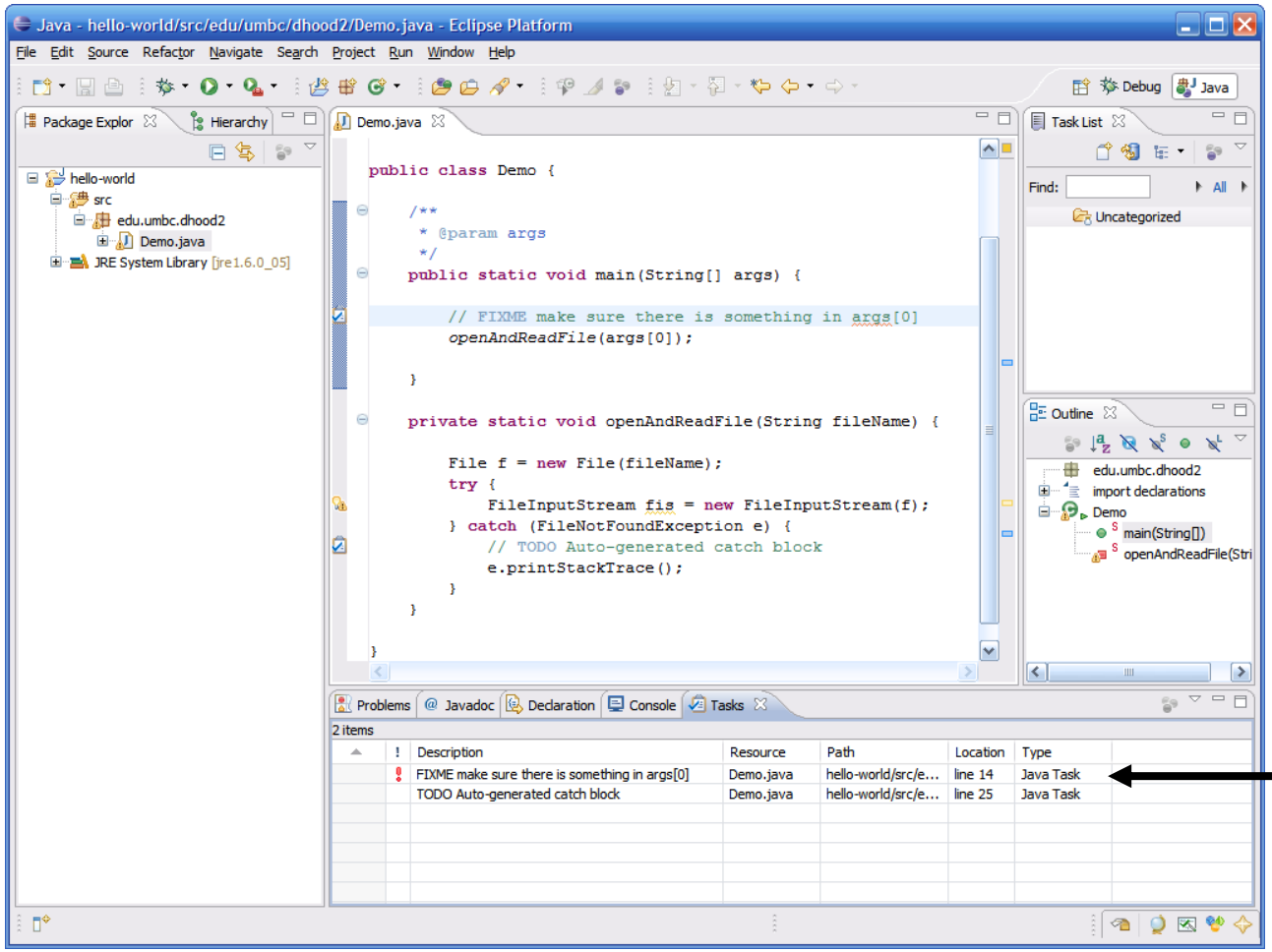
Tasks (continued)

- To add a table of all reminders in all of your source code you can add the Tasks view by clicking on Window → Show View → Tasks



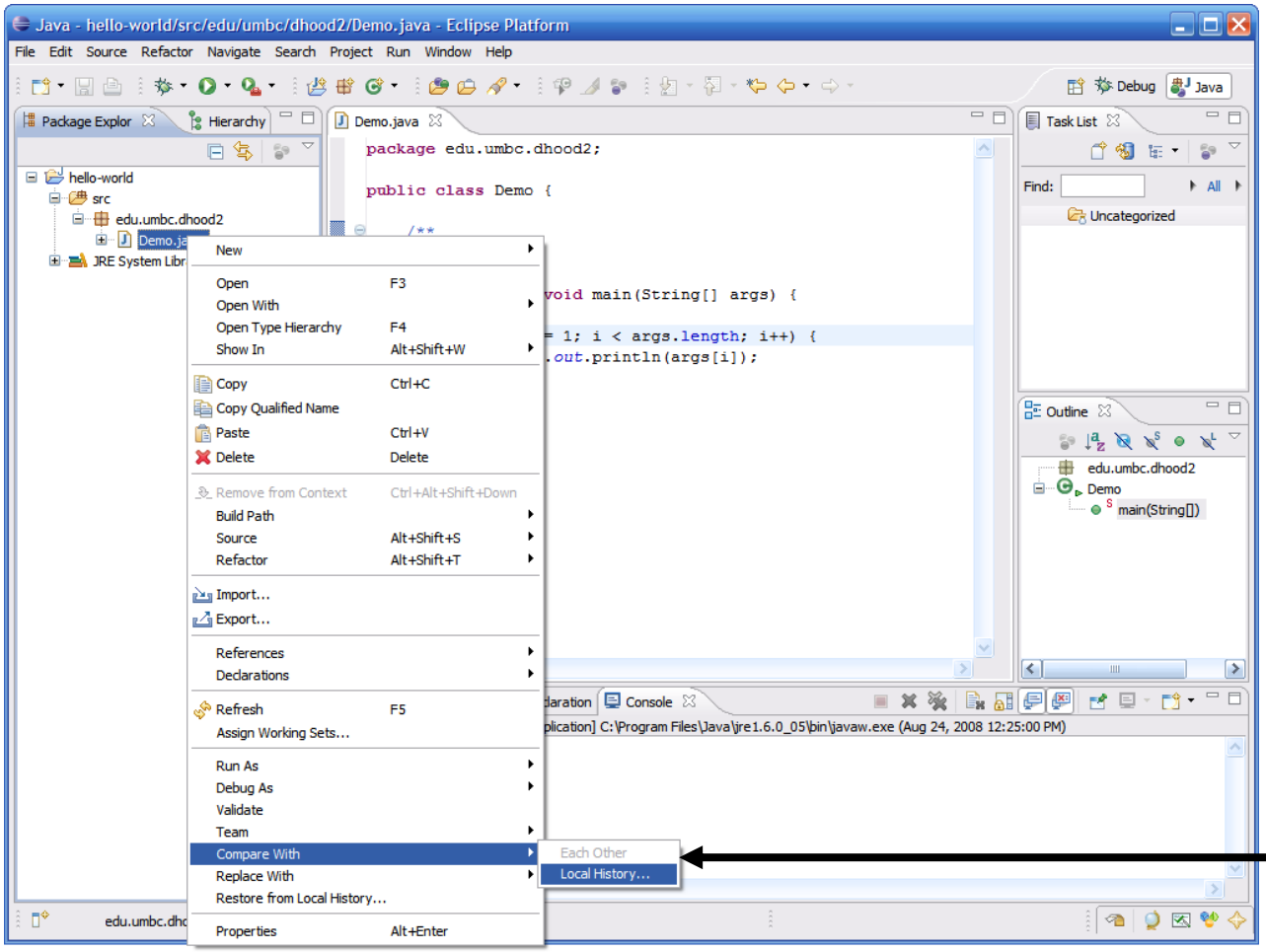
Tasks (continued)

- This neatly displays all tasks in a tabular form



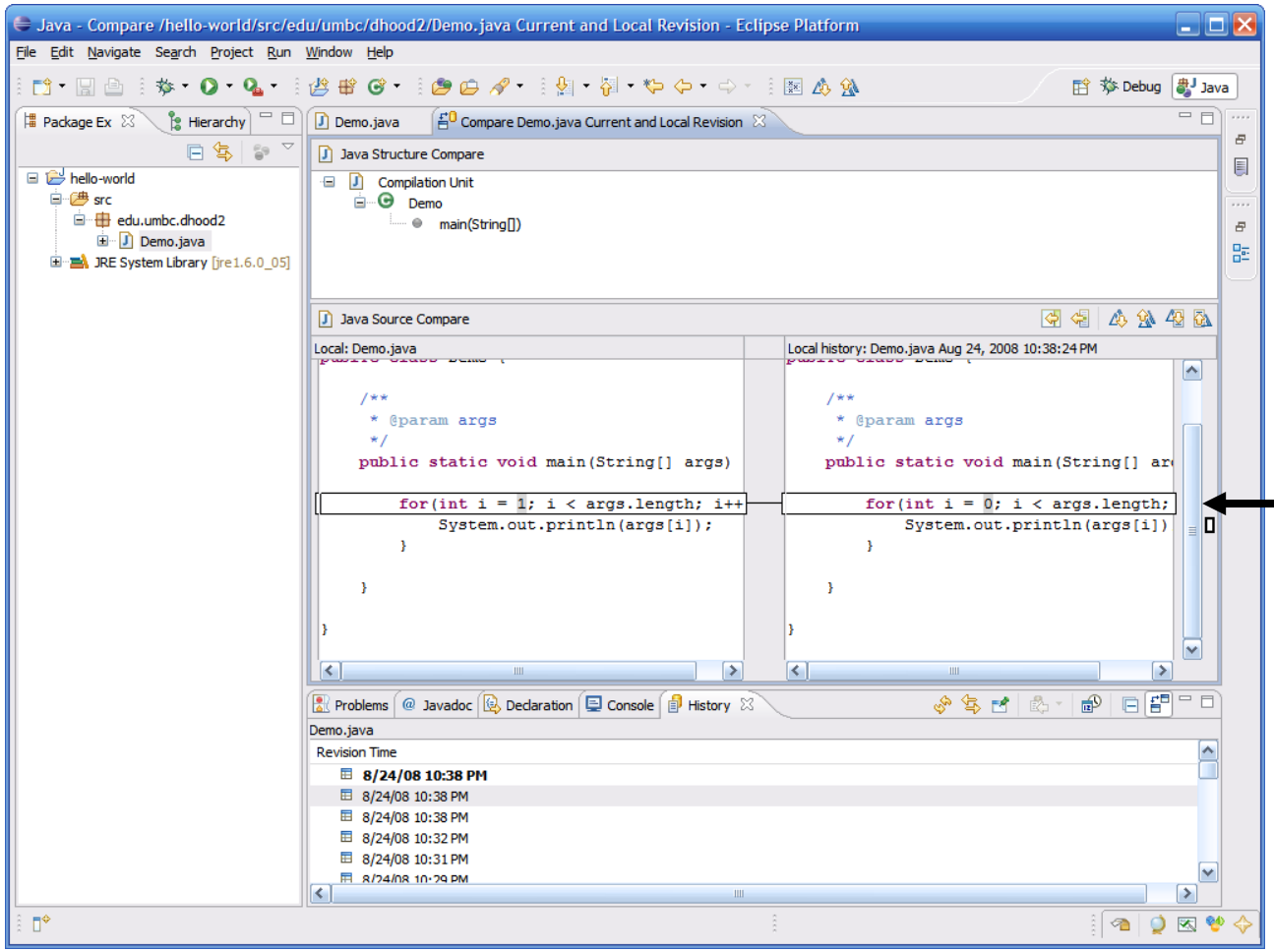
Local History

- Eclipse maintains a local history of file revisions which can be accessed by right clicking on the class, then selecting Compare With → Local History…



Local History (continued)

- Previous saved revisions are displayed in the History pane, double click a revision to view in the built-in diff viewer



Summary

- Benefits
  - Code completion
  - Faster code/compile/run cycles (real time)
  - Open source (free)
  - Extensible (plugins)
- Disadvantages
  - Pretty heavyweight
  - Requires JRE
  - Learning Curve