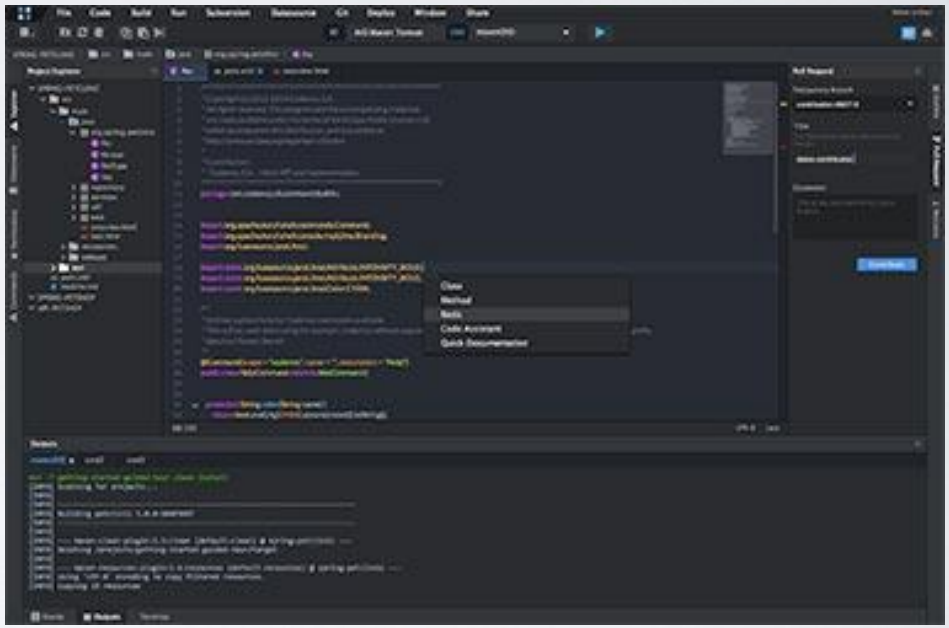
GETTING STARTED WITH ECLIPSE

Caitrin Armstrong

THE ECLIPSE IDE

- IDE = **I**ntegrated **D**evelopment **E**nvironment
- Language-neutral: Java, C, HTML, …
- Powerful, advanced features that help with code development (e.g. debugging, auto-completion).



DOWNLOADING ECLIPSE

- Version: Not a huge deal for our purposes, but why not just download the most recent!
- Eclipse “Oxygen” is the minimal IDE, suitable only as a base for installing other tools
  - So install the “Eclipse IDE for Java Developers”.
  - If you download just the base package then you will have to select this option anyways

DOWNLOADING ECLIPSE

Eclipse IDE for Java Developers

Package Description

The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Mylyn, Maven and Gradle integration

This package includes:

- Git integration for Eclipse
- Eclipse Java Development Tools
- Maven Integration for Eclipse
- Mylyn Task List
- Code Recommenders Tools for Java Developers
- Eclipse XML Editors and Tools

▸ Detailed features list

Download Links

Windows 32-bit
Windows 64-bit
Mac OS X (Cocoa) 64-bit
Linux 32-bit
Linux 64-bit

Downloaded 432,679 Times

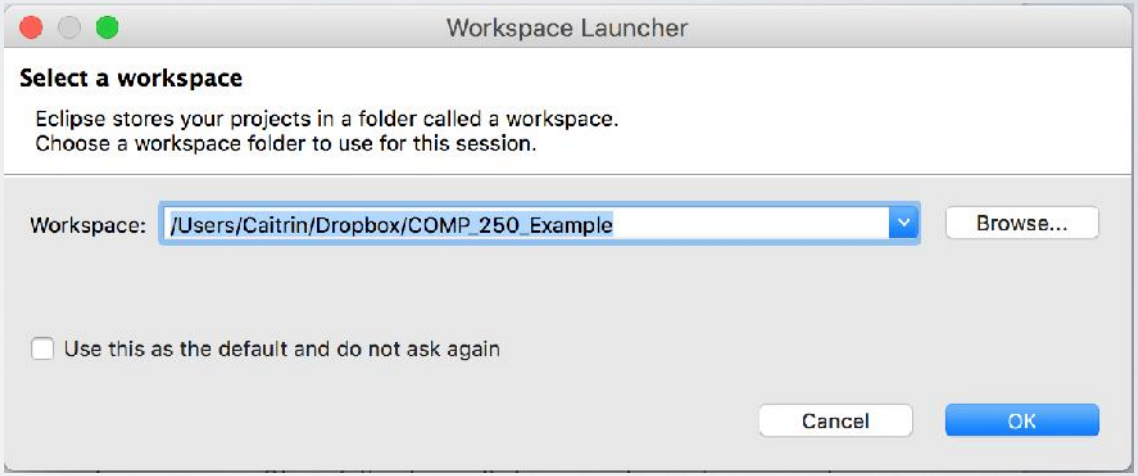
▸ Checksums...

Bugzilla

http://www.eclipse.org/downloads/packages/eclipse-ide-java-developers/oxygenr

ECLIPSE DIRECTORY STRUCTURE

- When you first open Eclipse you will be asked to specify a workspace. I recommend you place this inside the directory you've already created for COMP 250.

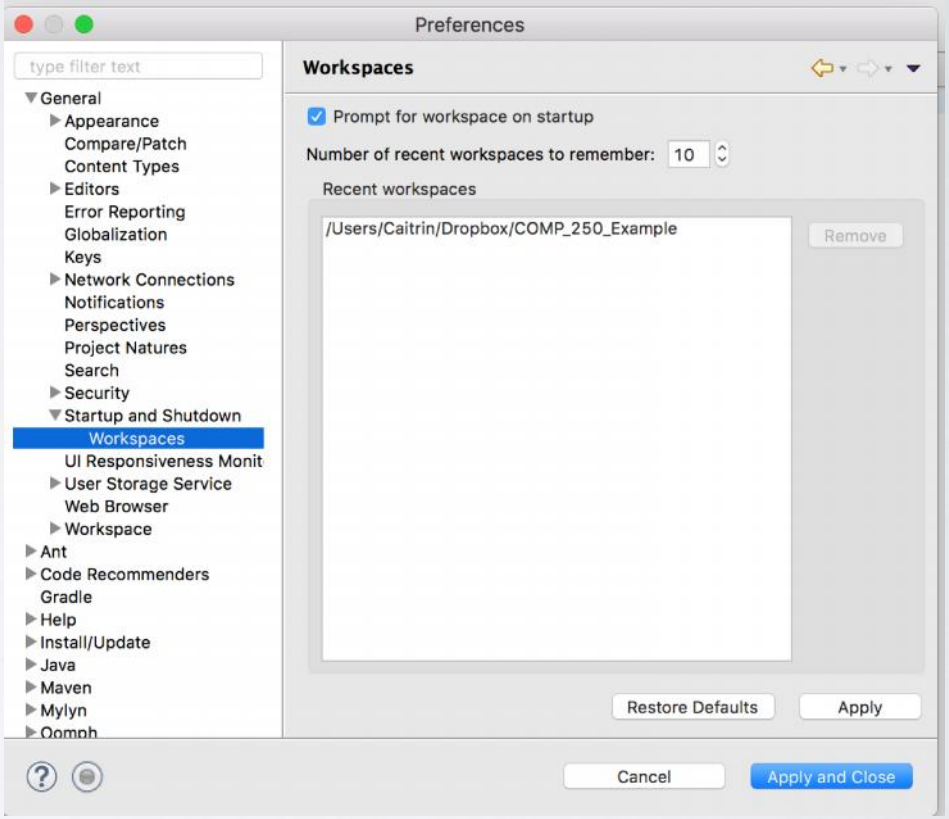


- Back up your work! Use some sort of versioning system: dropbox, office365. Your COMP 250 code directory will need to be a sub-directory of one of these systems' root directory.
  - If mac/linux: symlinks are a wondrous thing. Google them and learn to have your files (displayed) in two locations, one of them in your versioning software

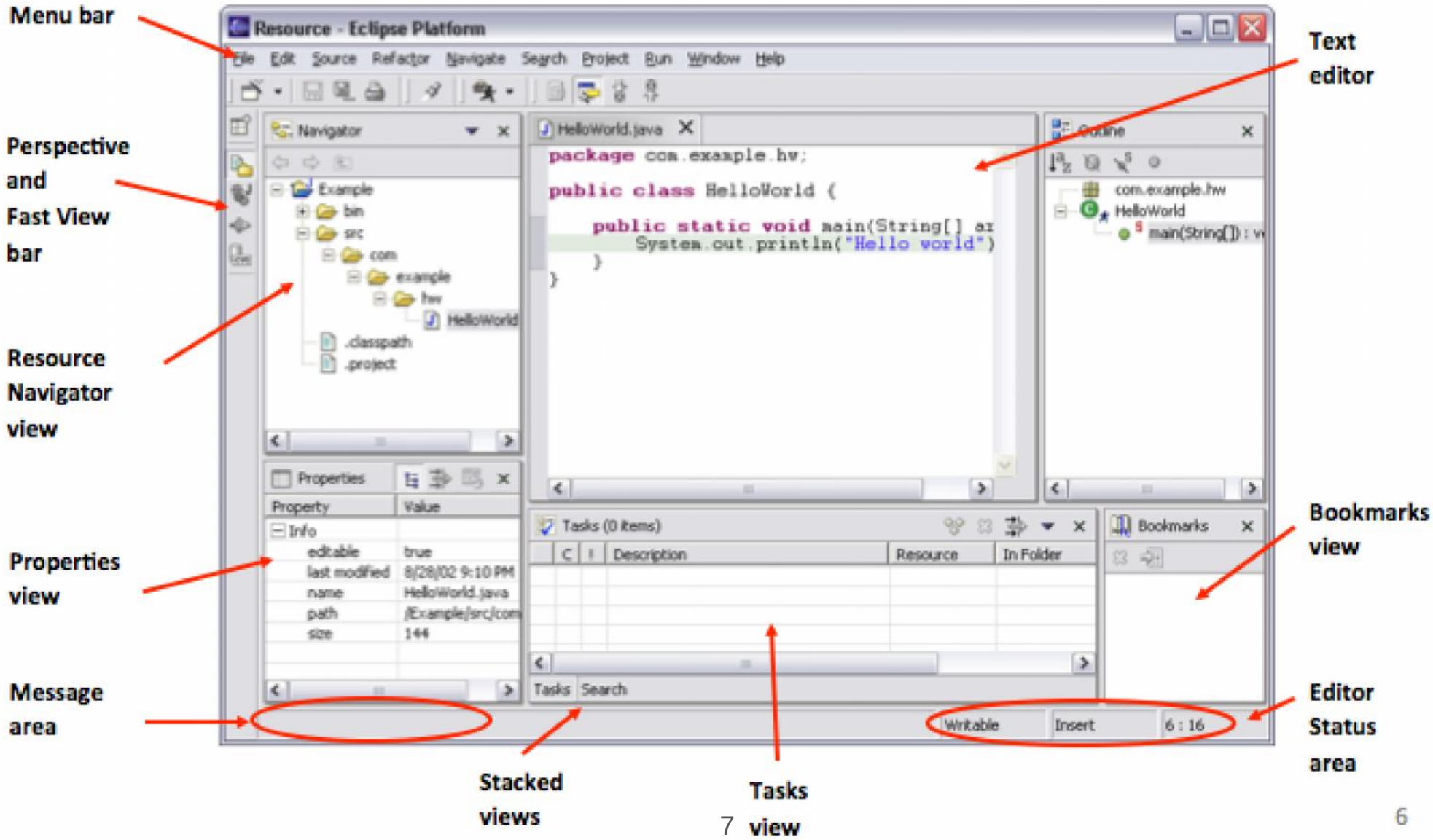
ALREADY DOWNLOADED?

- If you've already downloaded eclipse and clicked the box to set your default workspace, you can change this by: *preferences > general > startup and shutdown > workspaces*

- Set to be within Dropbox, Office 365 etc!



Workbench Terminology



FIRST STEPS: CREATING A NEW PROJECT

- For now you only need to supply the name and the directory location. Leave other options as-is.
- *File > New > Project*
- What is a project? It depends, perhaps a project is one course, perhaps you have a separate workspace for each course and assignments are projects.
  - See https://stackoverflow.com/questions/6310928/how-do-you-organize-100-projects-in-eclipse

FIRST STEPS: CREATING A NEW PACKAGE

- You must next create a package. A package contains all files for a specific purpose.
- *File* > *New* > *Package*
- A package name corresponds to a subdirectory within your project. If you change the package name, then you change the directory name.

FIRST STEPS: CREATING A NEW CLASS

- You should already understand the concept of a class!
- *File* > *New* > *Class*
- You can rename classes by right clicking on the class in the navigator view and selecting “refactor”. Eclipse will make all the necessary changes in your already-written code.

WRITING CODE: QUICK FIX

- Quick fix analyzes the document content for potential problems. You may see:
  - Errors highlighted by red squiggly lines
  - Warnings highlighted by yellow squiggly lines
  - Both errors and warnings displayed in the problem view
  - A light bulb in the vertical ruling indicating where there is a problem
- You can invoke a quick fix dialog by:
  - placing the mouse pointer on a squiggly line
  - clicking on the light bulb

WRITING CODE: CONTENT ASSIST

- Eclipse features content assist: press control - space to see a dialog class listing the class variables and methods. You can select one of the listed items.

- Very useful for writing complex code without having to search yourself for relevant information.

WRITING CODE: HOVER

- Hovering with the mouse pointer over a class being imported will display the java documentation (java doc) associated with the class. Java doc contains information on the class written by the developers.
- Hovering over a method shows the java doc for that method.

WRITING CODE - OTHER TIPS TO TRY AT HOME

- When the cursor is in a method argument, press Ctrl + Shift + Space to see a list of parameter hints.
- To add code around other code select a block of code and press Alt+Shift+Z to see a menu of items like if statement, for loop, try/catch etc that can enclose the selected block of code.
- Select an opening or closing bracket and press Ctrl+Shift+P to find its matching bracket.
- You can toggle line number visibility. Right click on the bar on the far left of the editor window.
- To comment out a block of code, select it and then go *Menu* > *Source* > *Toggle Comment*
- A useful refactoring is to mark code and create a method from the selected code. Highlight your code, right click on the selection and select *Refactoring* > *Extract Method*. You can then supply the name of the new method. You can also do this with a constant.

RUNNING CODE

- Right click on the java class that contains the main method
- Select run as > java application
- Or, click the green play button 

WHEN IT DOESN’T WORK: DEBUGGING CODE

- Debugging allows you to run a program interactively while watching the source code and the variables during the execution.
- To get to the debugger perspective, click the tiny bug in the top right corner.

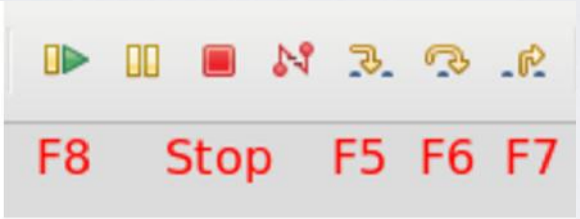
DEBUGGING CODE: BREAKPOINTS

- A breakpoint is used to specify where you want the program to stop while debugging. Once the program is stopped you can investigate variables and their content.
- Set a breakpoint by double clicking in the left margin on the line you want execution to stop on.
- After setting a breakpoint you can select the properties of the breakpoint by right clicking *Breakpoint Properties*.

DEBUGGING CODE: OTHER CONTROLS

- Set a method breakpoint by double-clicking in the left margin of the editor next to the method header.
- Set a class load breakpoint by right-clicking on a class in the Outline view and choose the Toggle Class Load Breakpoint option.
- Not only can we keep track of variables in a debugger, we can also keep track of particular expressions throughout the program using Watch Expression.

DEBUGGING CODE: BUTTONS



- F5 Executes the currently selected line and goes to the next line in your program. f the selected line contains a method call, then the debugger steps into the associated code.
- F6 steps over the call, i.e. it executes a method without stepping into it in the debugger.
- F7 steps out to the caller of the currently executed method. This finishes the execution of the current method and returns to the caller of this method.
- F8 tells the Eclipse debugger to resume the execution of the program code until is reaches the next breakpoint.

DEBUGGING CODE: VARIABLES VIEW

- The variables view displays fields and local variables from the current executing stack. This will appear once you have run the debugger.