1.00 Lecture 2

Interactive Development
Environment: Eclipse

Reading for next time: Big Java: sections 4.1-4.5

What's an IDE?

- **An integrated development environment (IDE) is an environment in which the user performs development tasks:**
    - **Creating and naming files to store a program**
    - **Writing code in Java or another language**
    - **Compiling code (checking <u>syntax</u> but <u>not logical</u> correctness, generating executable program)**
    - **Reviewing and testing the code with the debugger**
    - **And many other tasks: version control, projects, code generation, etc.**
- **Eclipse is a popular Java IDE**
    - **You must use it in 1.00 homework, lecture, and recitation**
   - **People write better software with an IDE**

What Does an IDE Do?

- **What does an IDE provide?**
  - Visual representation of program components
  - Ability to browse existing components easily, so you can find ones to reuse
  - Quick access to help and documentation to use existing libraries and tools versus writing your own
  - Better feedback and error messages when there are errors in your program
  - A debugger, which is <u>not</u> primarily used to debug, but is used to read and verify code
  - Communication between programmers in a team, who share a common view of the program
- **Your programs in 1.00 are small, but Eclipse will make life much easier**
  - In large projects, the benefits are greater still

Starting Eclipse

- **Start Eclipse by double clicking the icon on your desktop.**
- **Identify all the interface areas labeled on the next slide.**
  - The Main Window is the command center, holding menus, tabs, and buttons.
  - The Explorer allows you to manage files and sets of files (projects) that form programs.
  - The working area holds editor, compiler, outline, output or debugger windows as appropriate.
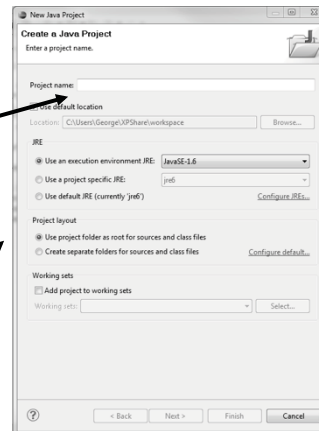
Anatomy of Eclipse

Main Window

Editor

Outline

Explorer

Console (output)
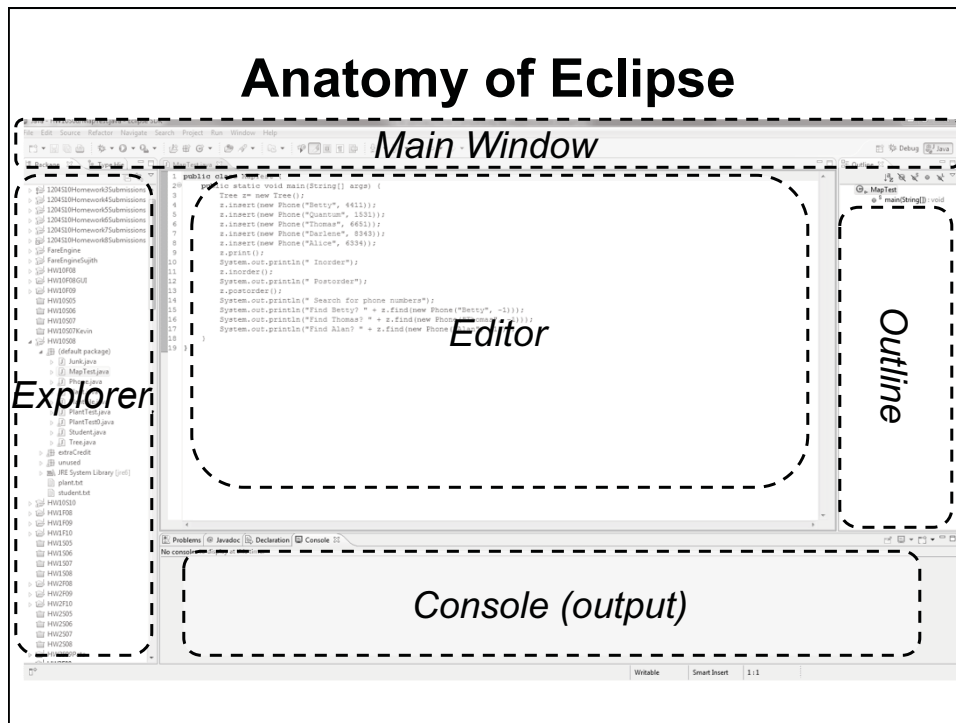
Courtesy of The Eclipse Foundation. Used with permission.

Creating a Project

Choose File-> New-> Java Project

A 'New Java Project'
  page appears

Project name: Lecture2

Make sure 'Use project folder' is checked
Your project folder will be in folder eclipse/workspace
Hit 'Finish'

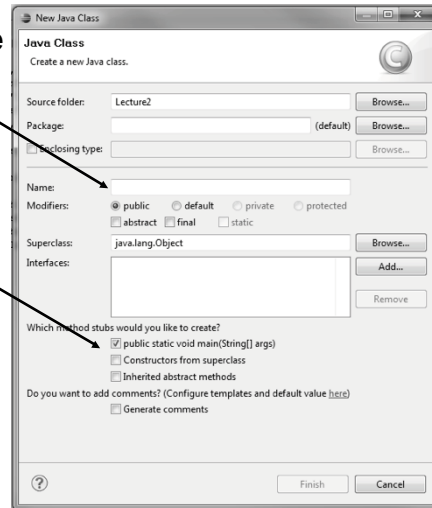Courtesy of The Eclipse Foundation. Used with permission.

Creating a Class

File-> New-> Class (or click 'New' icon)

Type class name: NauticalMile

Make sure 'public static void main…' is checked

Hit 'Finish'

main() is a method

Courtesy of The Eclipse Foundation. Used with permission.

The NauticalMile Program

- **A nautical mile is defined as the average length of a 1 minute arc of latitude on the earth's surface.**
- **The circumference of the earth is 24859.82 statute miles**
- **A statute mile contains 5280 feet**
- **The circumference is 360 degrees, and each degree contains 60 minutes**
- **Calculate the length of a nautical mile in feet as:**

$$nm = \frac{\text{number of feet in circumference}}{\text{number of minutes in circumference}}$$

- **Be careful about data types and division!**
- **Output your answer using System.out.println(…);**

NauticalMile.java

```
public class NauticalMile {
    public static void main( String[] args ) {
      double circum = 24859.82*5280;
      int minutesInCircle = 360*60;  // This is a comment
      double nautMile = circum / minutesInCircle;
      System.out.println(
          "Feet in a nautical mile = " + nautMile);
    }
  }  // Java is case sensitive
```

- **Write this Java program using Eclipse**
  - **Delete the Eclipse-generated comments at top**
- **Save it (ctrl-S or File->Save); Eclipse will compile it**
- **If you get any errors, fix them**
- **After it compiles, make some errors, experiment**

Compile Time Errors

- **Remove the semicolon from the end of the line that starts with**
  ```
  double circum
  ```
- **Move the mouse over the wavy line. You should see:**

  ```
  Syntax error, insert ";" to complete
  BlockStatements
  ```
- **There is also a red box on the right and a red circle on the left**
- **Fix the error**
- **Remove the semicolon from the next line**
  - **The error message is slightly different**

Running `NauticalMile` in Eclipse

- **Once you're able to save with no errors, select Run-> Run As-> Java Application**
- **Or use the green circle icon**
- **Save changes if prompted (OK)**
- **Part of working area may change from problem view to console view**

Neat Things About Eclipse

- **Key words are highlighted in purple.**
  - Strings are highlighted in blue
- **Click on a variable to see all occurrences in your file**
  - Refactor -> Rename if you want to change its name
- **Java classes have 'tool tips' that display info when you place your mouse over them (e.g., System)**
- **Eclipse will format your file**
  - Mess up the alignment of the text lines.
  - Then right click in the editor window and select
    `Source-> Format or Source-> Correct Indentation.`
  - Or use ctrl-A, ctrl-I.
- **Get full documentation of Java methods**
  - Place cursor on any built-in Java method or class
    - String or System, for example
  - Hit Navigate-> Open Attached Javadoc
- **Expand explorer view to see variables, methods**

Reading `NauticalMile`

- **Set a <u>breakpoint</u> to stop your program at or near its beginning**
  - **Right click on the left margin of the text editor at the desired line (`double circum= …`)**
  - **Select "Toggle Breakpoint"**
- **Select Run->Debug As -> Java Application**
  - **Or use the toolbar (bug icon) , but be careful what it runs**
- **Eclipse displays the Debug Perspective**
  - **Your program stops at the breakpoint line**

Eclipse Debug Perspective

Main Window

Status

Variables

Program code

Call stack

Console output
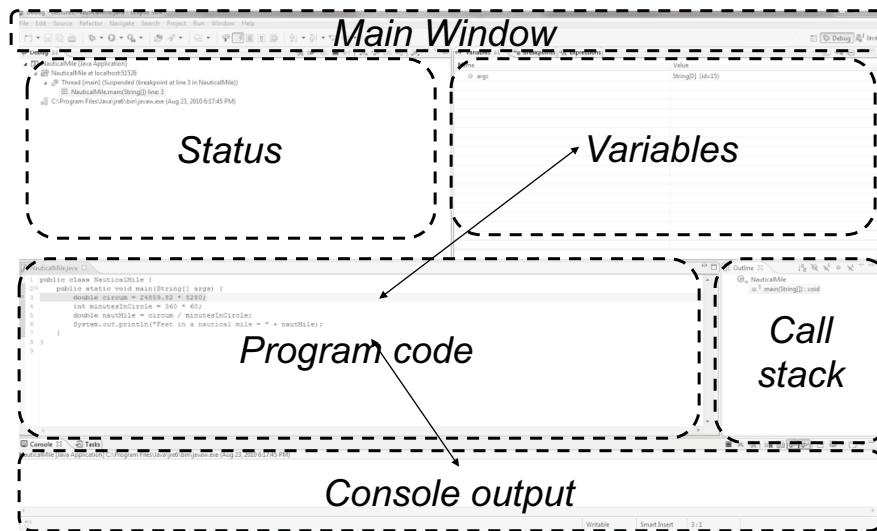
Courtesy of The Eclipse Foundation. Used with permission.

Stepping Through

- **Now step through NauticalMile line by line**
  - **Use the 'Step Over' icon or hit F6**

  - **Later we'll use 'Step Into' (F5) and 'Step Return' (F7)**
  - **We can use 'Resume' (F8) to run to the next breakpoint**
  - **And we use 'Terminate' to quit the program**
- **Variable values display in the Variables window**

Courtesy of The Eclipse Foundation. Used with permission.

Stepping Through, 2

- **The Step buttons are a functional family unit:**
  - **Step Into (F5) means stop at every line of code including every step of methods that are invoked.**
  - **Step Over (F6) means stop at every line of code in the current method but execute method calls in one step.**
  - **Step Return (F7) means run everything in the current method and stop when the method returns. In other words, run to the end of the method.**
  - **(All we have is a single main() method right now, but we'll have a lot more soon.)**
- **Click Step Over**

Examining Variable Values

- **In the top right frame of the Debugging View, you'll see the variables**
- **Click Step Over once more to advance another line.**
  - You should see that you just defined another variable, `minutesInCircle`.
- **Set another breakpoint at the last line (`System.out…`)**
- **Click the Resume button**   Terminate

- **The program stops at the last line.**
- **Click Resume or Step Over**
  - **The program output appears, and the program exits.**

Courtesy of The Eclipse Foundation. Used with permission.

Breakpoints

- **What if you are trying to figure out what is wrong with a homework program that's about 100 lines long?**
  - **Set a breakpoint at the beginning.**
  - **Run->Debug As->Java Application**
  - **Step Over line by line looking at variable values until you find an error**
  - **Go back to Java Perspective, fix the error, save the file**
    - **Don't fix it in Java Debug Perspective—less confusion**
  - **Set a breakpoint at the line you fixed**
  - **Run->Debug As-> Java Application (or toolbar icon)**
  - **The program will run to the line you fixed**
  - **Resume using Step Over from there**
- **You can right click and select 'Toggle Breakpoint' to get rid of unneeded ones**

Exiting the Debugger

- **Sometimes you want to exit the debugger without allowing your program to run to completion.**
- **Just click the Terminate button (red square) near the Resume button**
- **Occasionally you need to clean up the Status (Debug) window in the upper left frame**
  - **Right click in the Debug Window**
  - **Select Remove All Terminated**
  - **If something is still there, right click on it**
  - **Select Terminate and Remove**

Managing Files in a Project

- **Adding files:**
  - **Same as the first one: File->New Class and so on.**
- **Copying files:**
  - **Ctrl-C, Ctrl-V and give new name**
- **Deleting files:**
  - **Right click on file and delete**
- **Moving files:**
  - **Drag and drop**
- **Downloading files**
  - **Navigate to zip file, download to directory on laptop**
  - **Unzip the file in Download or 100 folder**
  - **Drag and drop the .java files into Eclipse browser**
- **Uploading files**
  - **Zip the .java files in the workspace folder, not .class files**
  - **Upload files. (Practice today, doesn't count.)**

Exercise

- **A bicyclist goes up a hill at 30 km/hr and comes down the same hill at 90 km/hr.**
- **Find and output the cyclist's average speed for this trip**
  - **It is not 60 km/hr**
- **Also find and output the average speed if the bicyclist goes up at 20 km/hr and comes down at 100 km/hr**
- **Before writing any code, make sure you understand the problem and can write the equation needed for the solution**
- **To use double values rather than int values, as this program requires, write all values as 1.0, 30.0, etc. rather than 1, 3, etc.**
- **File -> New-> Class -> Bicycle**
- **Write your code in the main() method**
- **Include comments that document your logic**
- **Save/compile and run your code. Step with the debugger.**

1.00 / 1.001 / 1.002 Introduction to Computers and Engineering Problem Solving
Spring 2012