

## **Class 3: The Eclipse IDE**

# **Introduction to Computation and Problem Solving**

**Prof. Steven R. Lerman  
and  
Dr. V. Judson Harward**

## **What is an IDE?**

- **An integrated development environment (IDE) is an environment in which the user performs the core development tasks:**
  - Naming and creating files to store a program
  - Writing code (in Java or another language)
  - Compiling code (checks correctness, generates an executable binary)
  - Debugging and testing the code
  - And many other tasks: version control, projects, code generation, etc.
- **Eclipse is one of the “standard” Java IDEs and the one that we will use in 1.00 for labs, tutorials and homework**

## Why Use an IDE?

- The alternative to an IDE is a 'command line' interface:
  - Not visual
  - Tools for each task are generally not tightly integrated
- 1.00 students in past terms have not used good software development practice with command line interfaces
  - Neither do many industry programmers
  - While you can do the same things with a command line interface as an IDE, in fact people don't
- People write software better with an IDE
  - The learning curve for an IDE is about the same as for command line tools, so it's what we use in 1.00

3

## What Do IDEs 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 on existing libraries and tools (vs writing your own)
- Better feedback and error messages when there are errors in your program
- A debugger, which is not 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

4

## Before Starting Eclipse

### Create a folder for your Java files

- In Windows Explorer, create folders:
  - Java
    - Lectures
      - Lecture3
- Put each homework, tutorial, and lecture projects each in a separate folder
  - Java thinks that files in the same directory are closely related

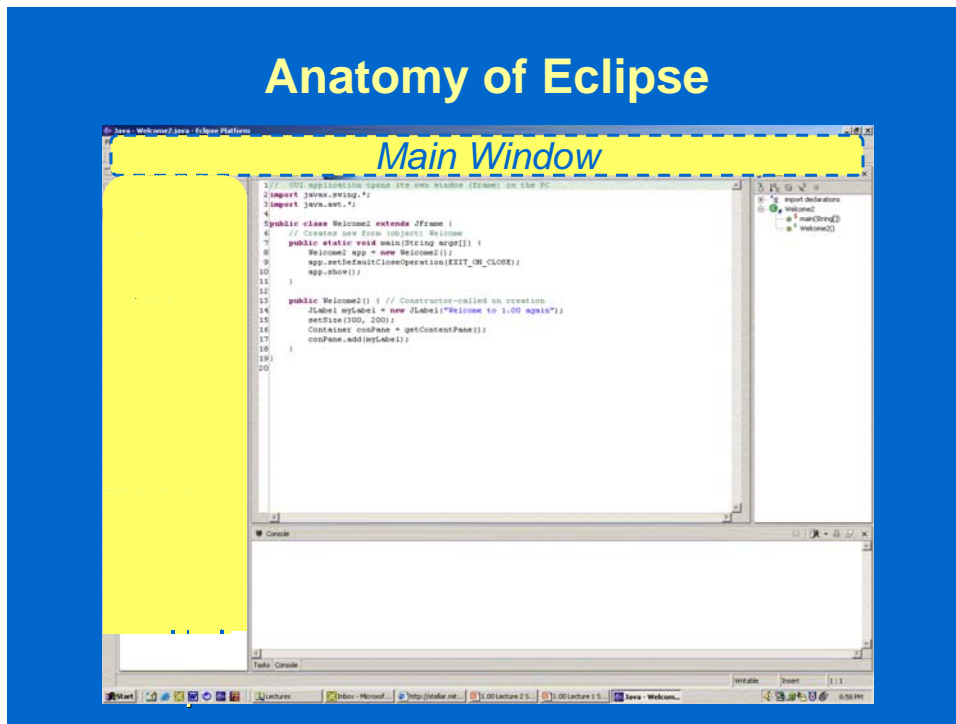
5

## 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, output or debugger windows as appropriate.
- If your display looks different, click the option to choose the “Java Perspective”

6

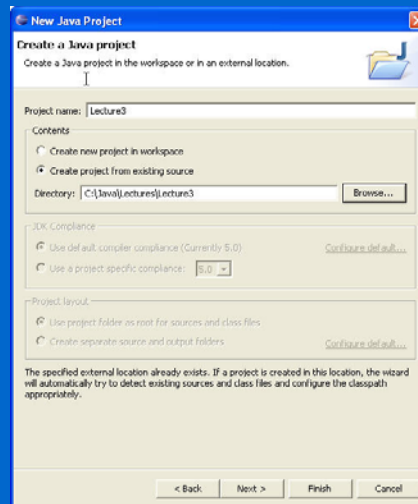
# Anatomy of Eclipse



## Creating a Project (2)

A 'New Java Project' page appears:

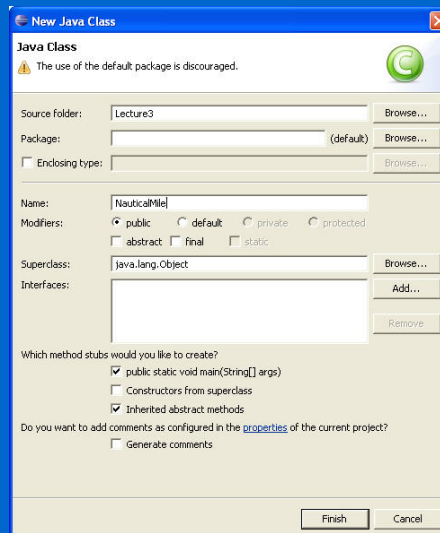
- Project name: Lecture3
- Check 'Create Project from existing source'
- Browse and select C:\Java\Lectures\Lecture3 using 'New Folder' button as necessary
- Hit 'Finish'



9

## Creating a Project (3)

- File->New->Class (or click 'New' icon)
- Type class name: NauticalMile
- Make sure 'public static void main(...)' is checked
- Hit 'Finish'



10

## The Nautical Mile Program

- A nautical mile is defined as the average length of a 1 minute arc of latitude on the earth's surface.
- So if I told you that the polar circumference of the earth was 24859.82 miles, you could calculate the length of a nautical mile in feet, right?

11

## NauticalMile.java

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

12

## Creating NauticalMile.java in Eclipse

- **Write NauticalMile.java in Eclipse**
  - Delete the Eclipse-generated comments at top
- **Save it (ctrl-S or File->Save); Eclipse will compile when you save**
- **If you get any errors, fix them!**
- **After it compiles, make some errors, experiment**

13

## Compile Time Errors

- Remove the semicolon from the end of the line that starts `double circum`
- Look in the Problems window at the bottom. You should see:  
`Syntax error, insert ";" to complete  
BlockStatements | NauticalMile.java | Lecture3 |  
line 6`  
with a wavy underline where the error was detected and an X at the margin.
- Click on error message and the corresponding line will be highlighted in the source file.
  - Fix the error.

14

## Running NauticalMile in Eclipse

- Once you're able to save with no errors, select Run->Run As-> Java Application
- Save changes if prompted (OK)
- Working area changes from editor to output view with console output at bottom of page.

15

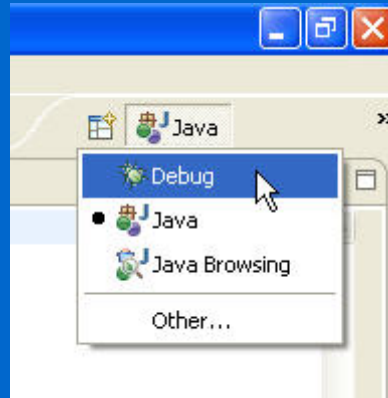
## Neat Things About Eclipse

- Key words are highlighted.
- Java built-in classes have 'tool tips' that show when you place your mouse over them.
- Type into the window to mess up the alignment of the text lines. Then right click in the editor window and select Source->Format. It will realign your margins.
- Get full documentation of Java methods:
  - Place cursor on any method or class
  - Hit Navigate-> Open External Javadoc
  - If you don't have this enabled, we'll do it at the end of class

16

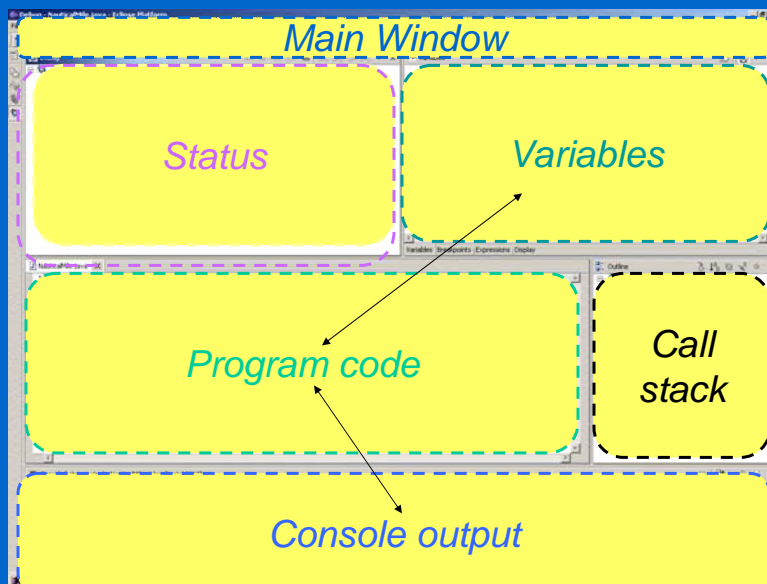
## Checking Your Program

- Select Debug from the “Perspective menu in the upper right corner
- The “Debug Perspective” appears, as shown on the next page (you had been in the “Java Perspective” when writing the program)
- You’ll use this a lot to read and correct (debug) your programs



17

## Eclipse Debug Perspective



## Reading Nautical Mile

- Set a breakpoint 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”
  - Or simply double click on the left margin
- Select Run->Debug As->Java Application
- Eclipse displays the Debug Perspective
  - Your program stops at the breakpoint line

19

## Stepping Through

- Now step through NauticalMile line by line
  - Use the ‘Step Over’ icon or hit F6



- (Later we'll use ‘Step Into’ and ‘Step Return’)
- Variable values display in the Variables Window

20

## Stepping Through, 2

- The Step buttons are a functional family unit:
  - Step Into means stop at every line of code including following method calls.
  - Step Over means stop at every line of code in the current method but execute method calls in one step.
  - Step Return means hurdle over everything in the current method and stop when the method returns.
- Click Step Over

21

## 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



- The program stops at the last line.
- Click Continue
- The program output appears, and the program exits.

22

## 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
  - Terminate and go back to Java Perspective, fix the error, save the file
  - Set a breakpoint at the line you fixed
  - Run->Debug Last Launched
  - The program will run to the line you fixed
  - Resume using Step Over from there
- You can right click and select 'Remove Breakpoint' to get rid of unneeded ones

23

## Exiting the Debugger

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

24

## Managing Files in a Project

- **Deleting files:**
  - Go back to the Java Perspective, and right click `NauticalMile`. Selecting `Delete` will erase the file. (But hold onto it for now.)
- **Adding files:**
  - Same as the first one: `File->New Class` and so on.
  - Later in the term you'll add other kinds of files to a project.

25

## Definition of the Meter

- The French originally defined the meter to be  $1/40,000,000$  of polar circumference of the earth.
  - A kilometer is  $1/40,000$  of polar circumference
- Using that fact and whatever you want to cut and paste from `NauticalMile`, create the code in class `Kilometer` to calculate how many kilometers there are in a nautical mile and print it to the Output Window.
  - Since you need decimal arithmetic, define variables as `double`
  - Or cast (convert) `int` to `double` as in Lecture 1 notes
- **Compile and debug.** Raise your hand if you need help. To execute `Kilometer` rather than `NauticalMile` the first time, from the Java view right click `Kilometer` in the Package Explorer and select `Run` (or `Debug`).
- If you get 1.609, that's the wrong answer.

26

## Attaching Javadoc

- In the Eclipse menu bar, go to
  - 'Window'->'Preferences'->'Java'->'Installed JREs'.
- There should be only one installed JRE (jre1.5.0\_04)
- Highlight it and click 'Edit...'.
  - Make sure that "Use default system libraries" is unchecked.
  - Browse for the correct folder ('C:\Program Files\Java\jre1.5.0\_04\docs\api')
  - Click OK.
- Javadoc is now linked to Eclipse.
- In Eclipse:
  - Place the cursor on any builtin Java method or class,
  - Select 'Navigate'->'Open External Javadoc' (or Shift+F2)
  - You now have full documentation on the Java class or method
- Ask a TA for help if you have questions