1.00 Lecture 2

Interactive Development Environment: Eclipse

Reading for next time: Big Java: sections 3.1-3.9
(Pretend the method is main() in each example)

What's 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 (check correctness, generate executable program)**
  - **Debugging and testing the code**
  - **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, tutorial**
  - **People write better software with an IDE**

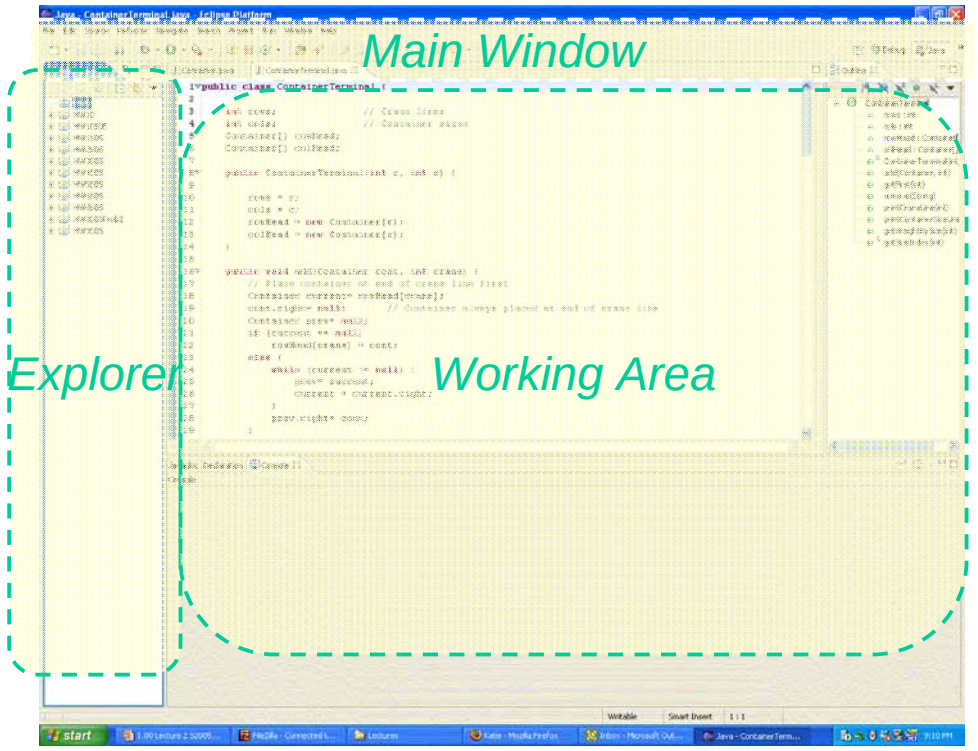
What Do IDEs Do?

- **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, documentation to use 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**
- **Your programs in 1.00 are small, but Eclipse will make life much easier**
  - **In large projects, the benefits are greater still**

Starting Eclipse

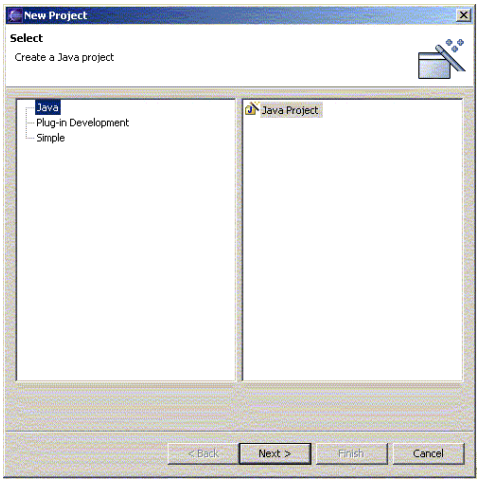
- **Start Eclipse by double clicking the icon on your desktop.**
- **Identify all the interface areas labeled on the next slide.**
  - **The Main Window is the command center, holding menus, tabs, and buttons.**
  - **The Explorer allows you to manage files and sets of files (projects) that form programs.**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

Anatomy of Eclipse



Creating a Project

Choose File->New->Project

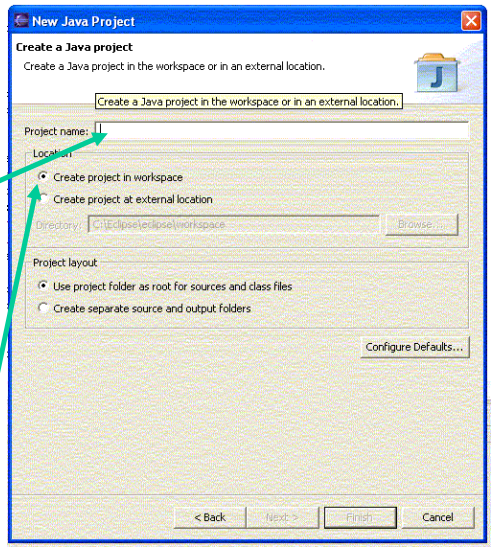


Make sure ‘Java’ is highlighted, then click ‘Next’

Creating a Project (2)

A ‘New Java Project’ page appears

Project name: Lecture2



Make sure ‘Create project in workspace’ is checked
Your project folder will be in folder eclipse/workspace
Hit ‘Finish’

Creating a Class

File->New->Class (or click ‘New’ icon)
Select ‘Class’, hit ‘Next’
Type class name: NauticalMile
Make sure ‘public static void main…’ is checked
Hit ‘Finish’

main() is a method

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);
   }
 }
```

- Write this program in Eclipse
  - Delete the Eclipse-generated comments at top
- Save it (ctrl-S or File->Save); Eclipse will compile
- 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**

```
double circum
```

- **Look in the Tasks window at the bottom. You should see:**

```
Syntax error on keyword "int"; ";" expected
    NauticalMile.java Lecture2    line 5
```

  with a wavy line 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.**

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**

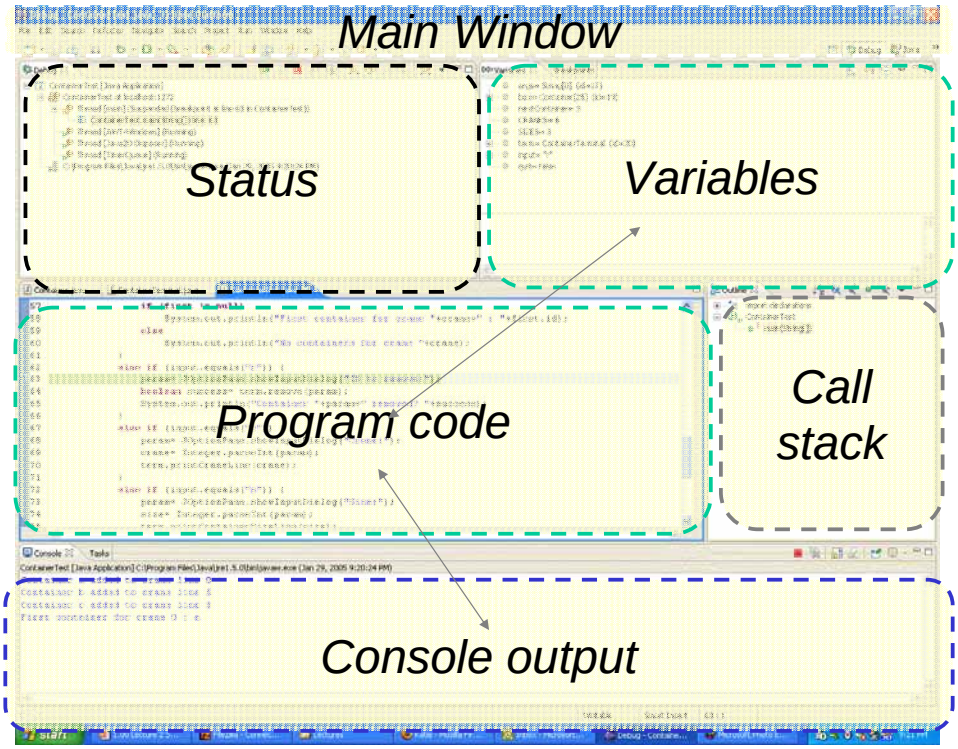
Neat Things About Eclipse

- 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, install Javadoc

Reading Your Program

- Select Run->Debug As-> Java Application
- Your `NauticalMile` program will run to completion and now the "Debug Perspective" displays, 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

Eclipse Debug Perspective

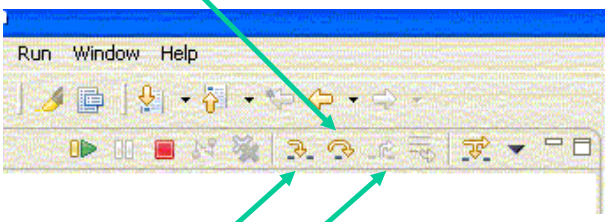


Reading `NauticalMile`

- **Go back to Java Perspective**
  - **Window->Open Perspective->Java**
  - **Or use the icon on the upper left margin**
- **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 "Add Breakpoint"**
- **Select Run->Debug Last Launched**
  - **Or Run->Debug As->Java Application, as before**
- **Eclipse displays the Debug Perspective**
  - **Your program stops at the breakpoint line**

Stepping Through

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



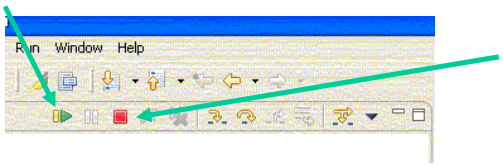
  - (Later we'll use 'Step Into' and 'Step Return')
- Variable values display in the Variables window

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 run everything in the current method and stop when the method returns.
  - (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



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

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
  - 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

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 (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

- **Deleting files:**
  - **Go back to the Java Perspective, and right click NauticalMile. Select Delete.**
- **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.**

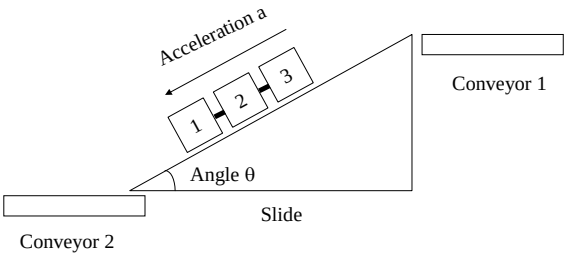
Exercise

- Compute the number of nautical miles in a statute (5280 foot) mile
- To do this:
  - Create a new class called `MileConvert` in the same project
  - Using whatever you want to cut and paste from `NauticalMile`, create the code to calculate how many miles there are in a nautical mile and print it to the Output Window.
    - Since you need decimal arithmetic, define variables as `double`
    - More on data types in Lecture 3!
  - Compile and debug. Raise your hand if you need help.
  - If you have time, compute the number of statute miles in a nautical mile and print it.

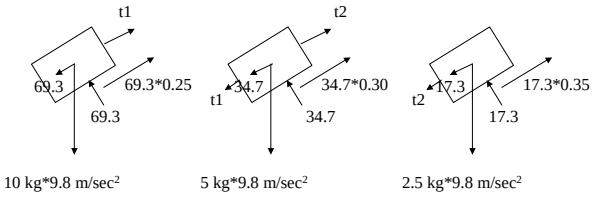
Attaching Javadoc

- In the Eclipse menu bar, go to
  - 'Window'->'Preferences'->'Java'->'Installed JREs'.
- There should be only one installed JRE (j2re1.5.0)
- Highlight it and click 'Edit...'.
  - You will see the field for 'Javadoc URL'.
  - Browse for the correct folder ('C:Program Files¥ Java¥j2sdk1.5.0¥docs¥api¥')
  - Click OK.
- Javadoc is now linked to Eclipse.
- In Eclipse:
  - Place the cursor on any 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

Homework 1



Homework 1, cont.



Free body diagrams at θ = 45°

Choose a small angle theta
Find the acceleration a and tensions t1, t2 from three equations
Find velocity v at the end of the slide length s
Increase the angle until velocity exceeds maximum (iteration)