



COMP 110/401

APPENDIX: INSTALLING AND USING ECLIPSE

Instructor: Prasun Dewan (FB 150, dewan@unc.edu)

SHORTCUTS (MSFT)/COMMANDS

- Block quote/unquote (CTRL-/))
- Run project (CTRL-F11)
- Search project for strings (Select project, Search→File)
- Step into, over, return (F5, F6, F7)
- Copy a project (CTRL-C, CTRL-V)
- Copy a package, class file to another project (CTRL_C, CTRL-V)
- Show variable, class, interface (identifier) definition (click, F3 or Right Menu→ Open Declaration) and return back (ALT ←)
- Find identifier (variable, class, interface) use (Click, CTRL-SHIFT-G, Right Menu→References)
- Refactor (Click, CTRL-SHIFT-R or Right Menu→Refactor)
- Complete identifier name (CTRL-Space)
- Automatic imports (CTRL-SHIFT-O)
- JUnit
- Change Method Signature (Click ALT_SHIFT-C, Right Menu→Refactor)
- Extract interface (Click ALT_SHIFT-C, Right Menu→Refactor)

CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface
- Supply main argument
- Create project from existing code

CONTENTS

- **Install JDK**
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface
- Supply main argument
- Create project from existing code

FIND JDK

<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

Java SE 8u144 Java SE 8u144 includes important bug fixes. Oracle strongly recommends that all Java SE 8 users upgrade to this release. Learn more ▶	
<ul style="list-style-type: none">▪ Installation Instructions▪ Release Notes▪ Oracle License▪ Java SE Products	JDK DOWNLOAD ⬇ Server IDE

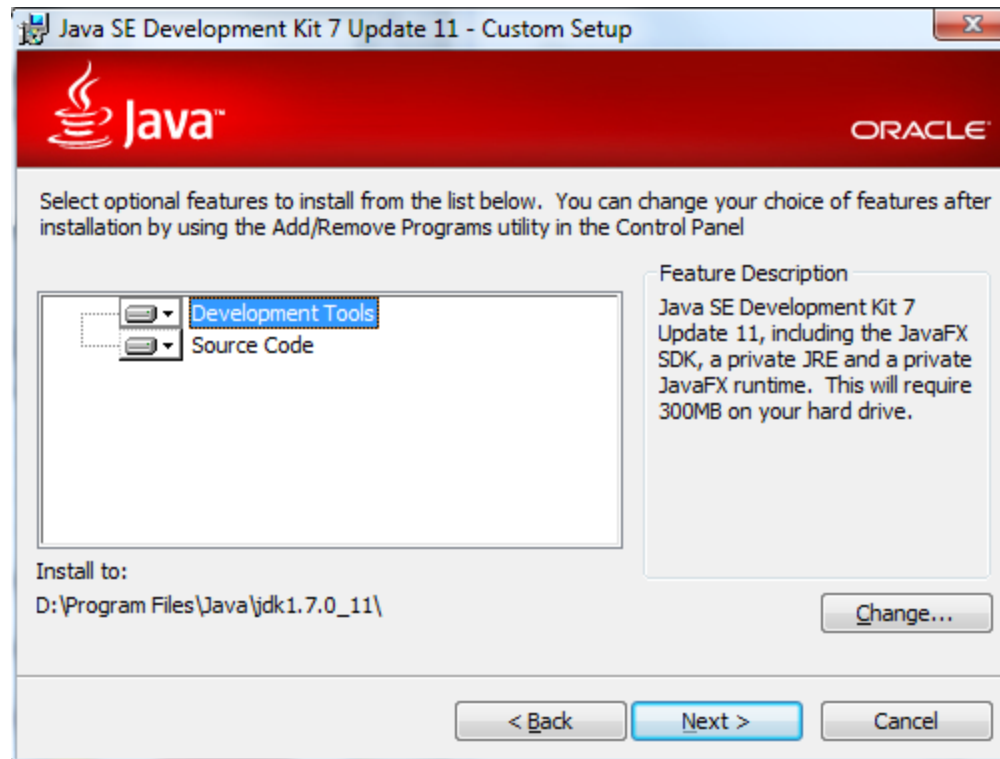
Check with instructor which JDK version to get

FOLLOW DOWNLOAD LINK: ACCEPT LICENSE AGREEMENT, FIND JDK FOR APPROPRIATE PLATFORM, AND CLICK ON EXE LINK TO DOWNLOAD EXE IN SOME TEMPORARY FOLDER

Java SE Development Kit 8u144		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
<input type="radio"/> Accept License Agreement <input type="radio"/> Decline License Agreement		
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	77.89 MB	jdk-8u144-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	74.83 MB	jdk-8u144-linux-arm64-vfp-hflt.tar.gz
Linux x86	164.65 MB	jdk-8u144-linux-i586.rpm
Linux x86	179.44 MB	jdk-8u144-linux-i586.tar.gz
Linux x64	162.1 MB	jdk-8u144-linux-x64.rpm
Linux x64	176.92 MB	jdk-8u144-linux-x64.tar.gz
Mac OS X	226.6 MB	jdk-8u144-macosx-x64.dmg
Solaris SPARC 64-bit	139.87 MB	jdk-8u144-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	99.18 MB	jdk-8u144-solaris-sparcv9.tar.gz
Solaris x64	140.51 MB	jdk-8u144-solaris-x64.tar.Z
Solaris x64	96.99 MB	jdk-8u144-solaris-x64.tar.gz
Windows x86	190.94 MB	jdk-8u144-windows-i586.exe
Windows x64	197.78 MB	jdk-8u144-windows-x64.exe

Java SE Development Kit 7u11	
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.	
<input checked="" type="radio"/> Accept License Agreement <input type="radio"/> Decline License Agreement	

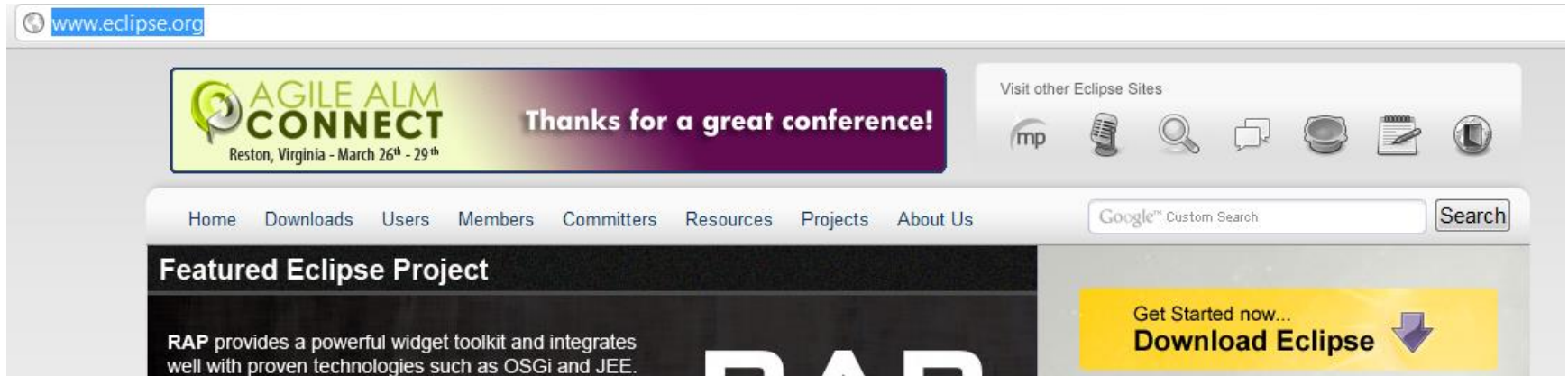
RUN EXE AND CHOOSE JDK FOLDER



CONTENTS

- Install JDK
- **Install Eclipse**
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

DOWNLOAD FROM [WWW.ECLIPSE.ORG](http://www.eclipse.org) (WINDOWS)



Eclipse Classic 3.7.2, 174 MB

Downloaded 1,236,277 Times

[Details](#)

[Other Downloads](#)




[Windows 32 Bit](#)

[Windows 64 Bit](#)

Download eclipse-java-indigo-SR2-win32.zip from:



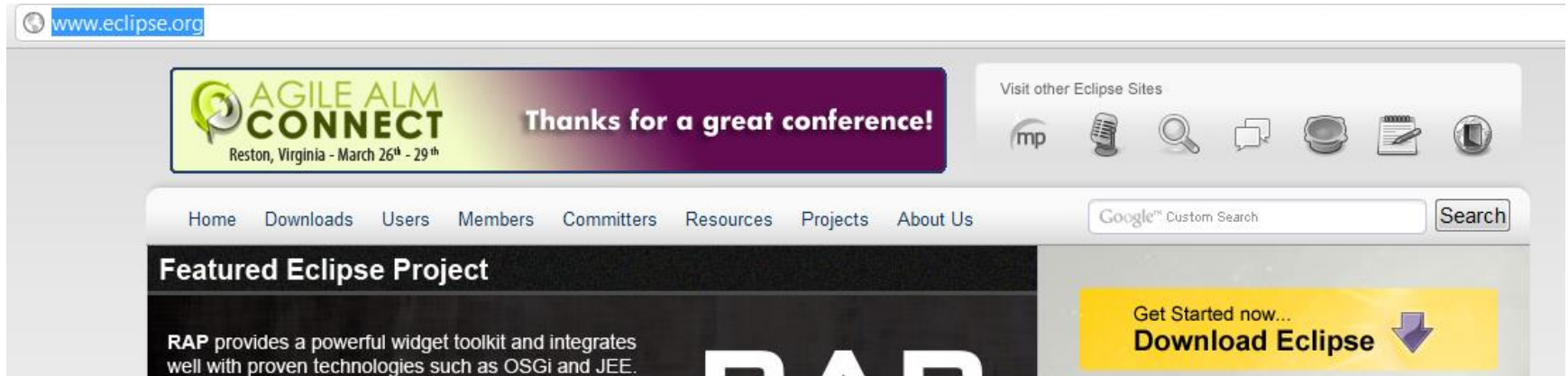
[\[United States\] Columbia University \(http\)](#)

Checksums: [\[MD5\]](#) [\[SHA1\]](#)  [BitTorrent](#)

...or pick a mirror site below.

Get the latest IDE for Java
Developers, not Indigo

DOWNLOAD FROM [WWW.ECLIPSE.ORG](http://www.eclipse.org) (WINDOWS)



Eclipse Classic 3.7.2, 174 MB

Downloaded 1,236,277 Times

[Details](#)

[Other Downloads](#)




[Windows 32 Bit](#)

[Windows 64 Bit](#)

Download eclipse-java-indigo-SR2-win32.zip from:



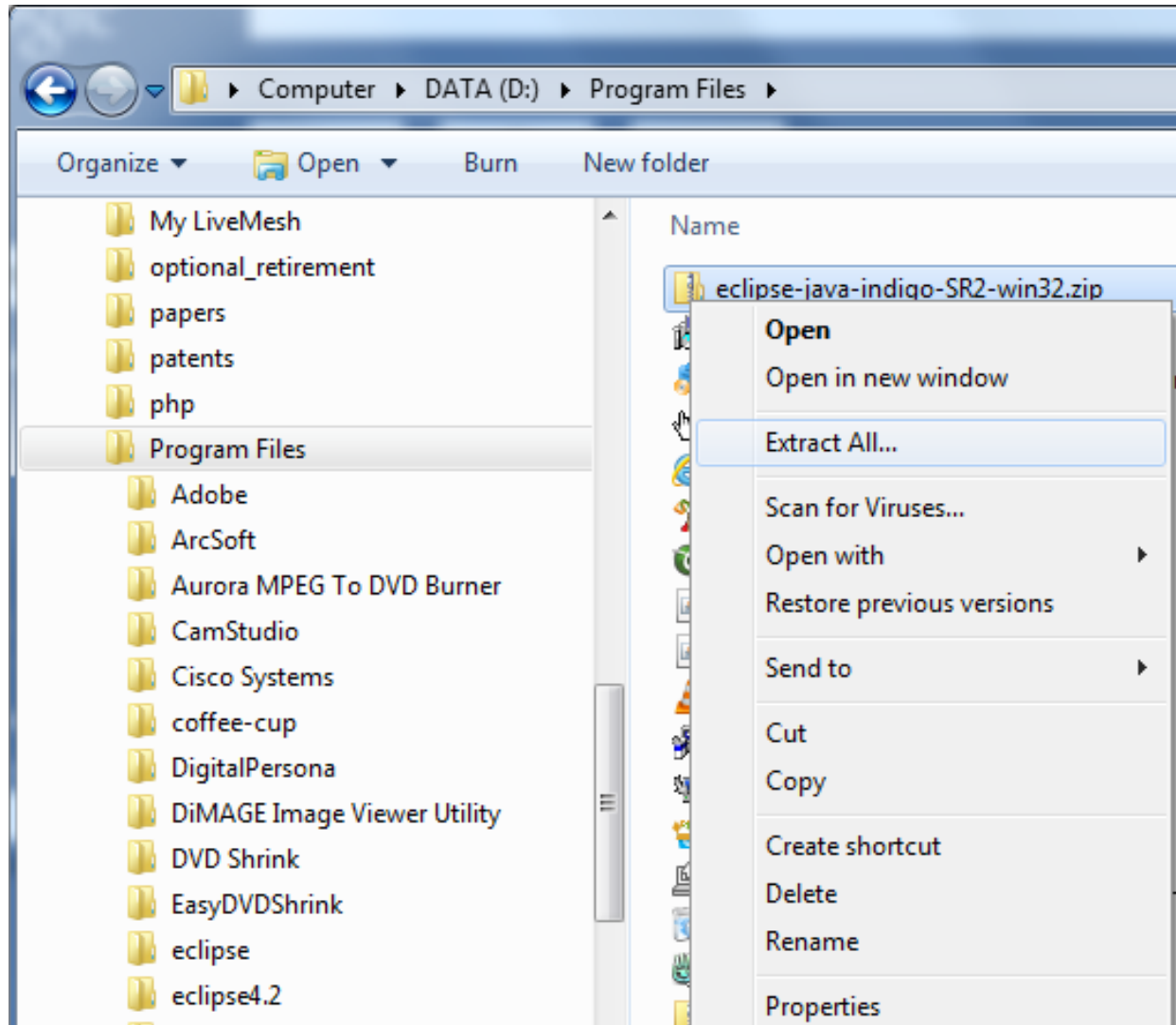
[\[United States\] Columbia University \(http\)](#)

Checksums: [\[MD5\]](#) [\[SHA1\]](#)  [BitTorrent](#)

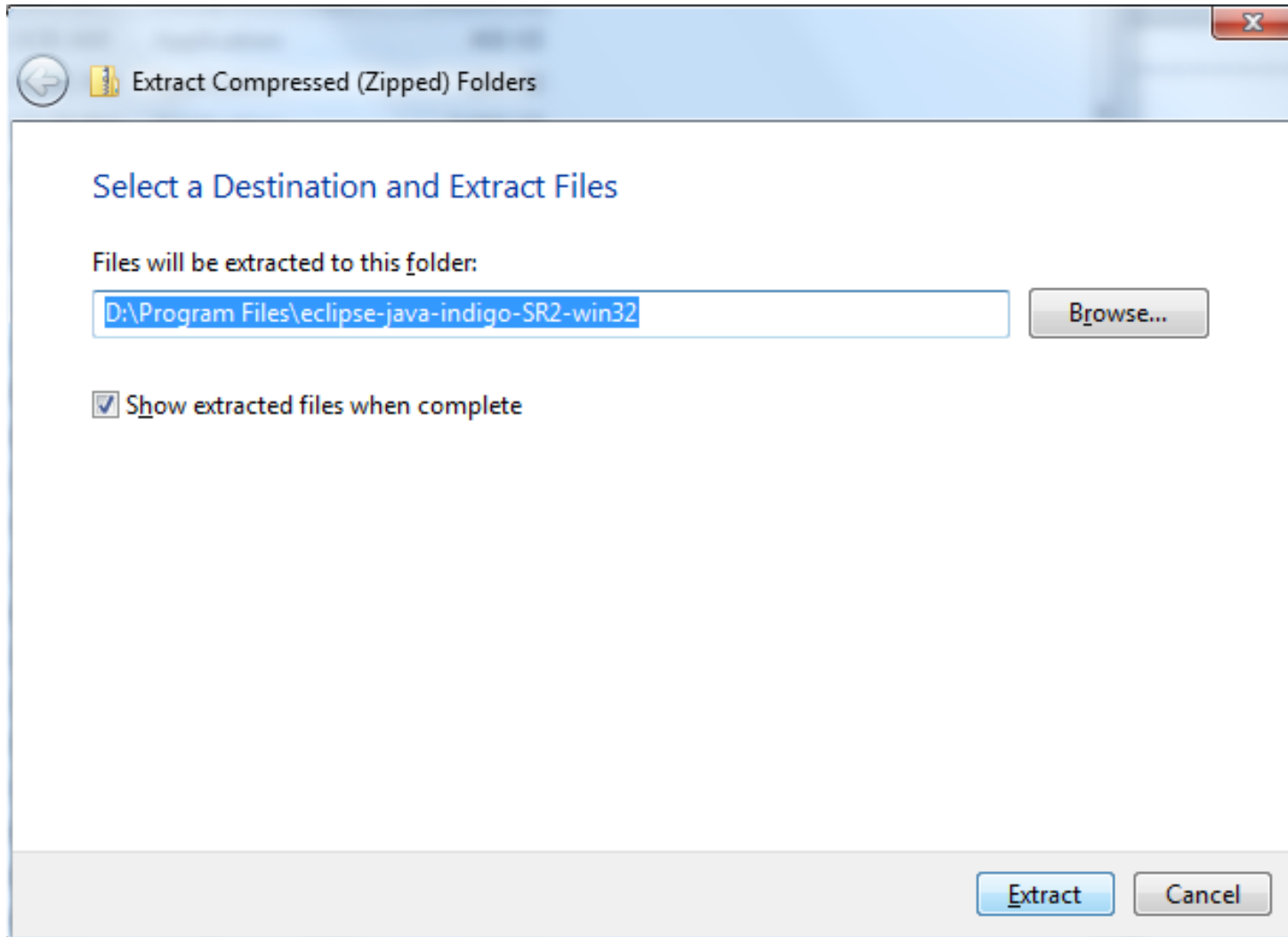
...or pick a mirror site below.

Get the latest IDE for Java
Developers, not Indigo

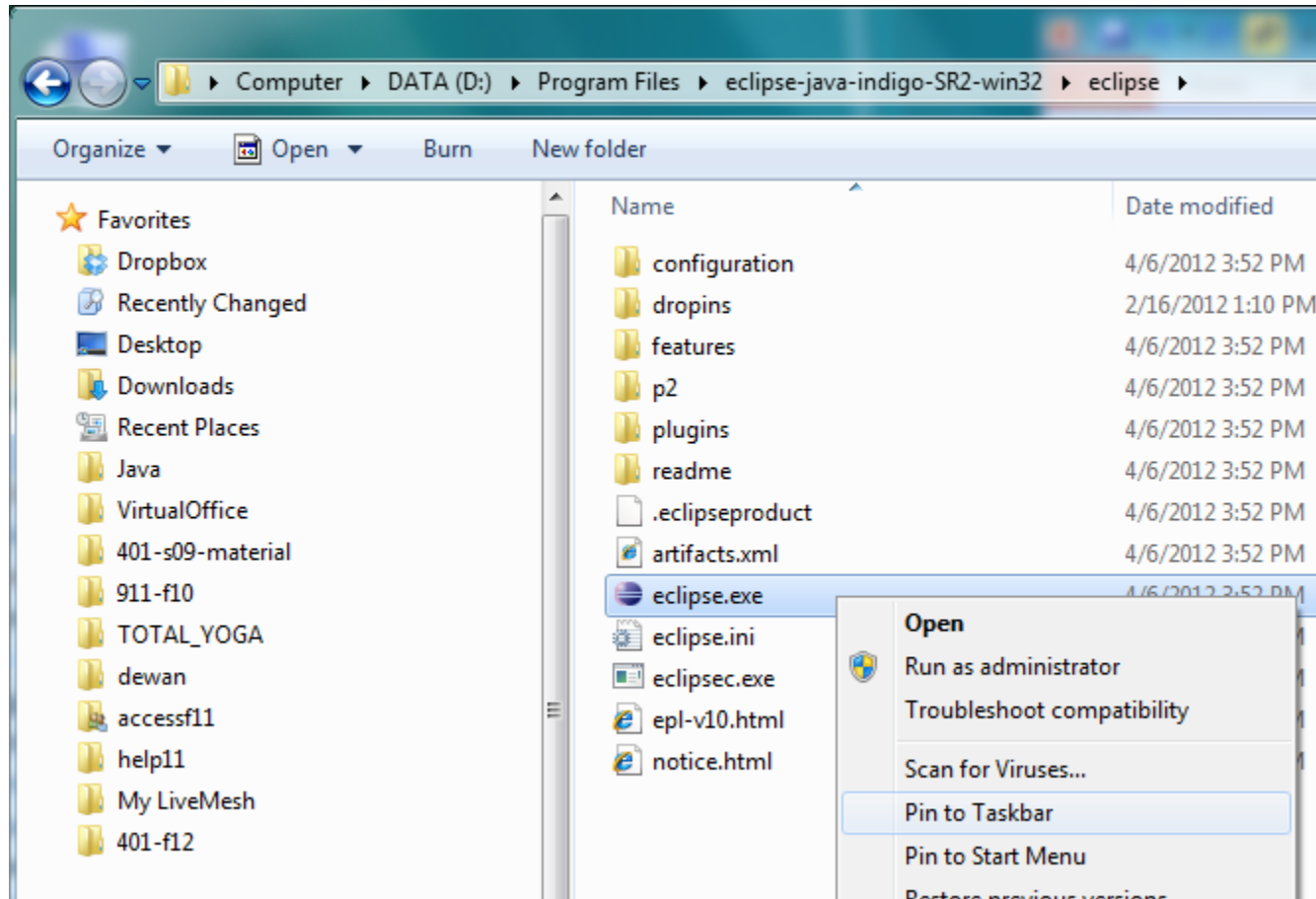
EXTRACT ALL



SELECT DESTINATION FOLDER



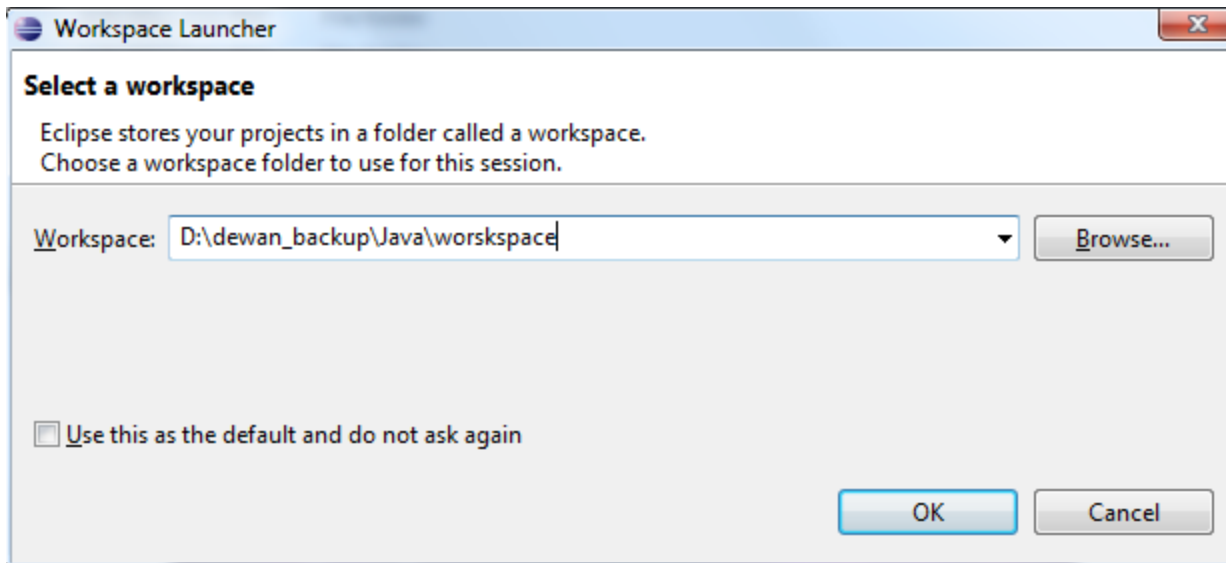
CREATE SOME KIND OF SHORTCUT FOR ECLIPSE.EXE



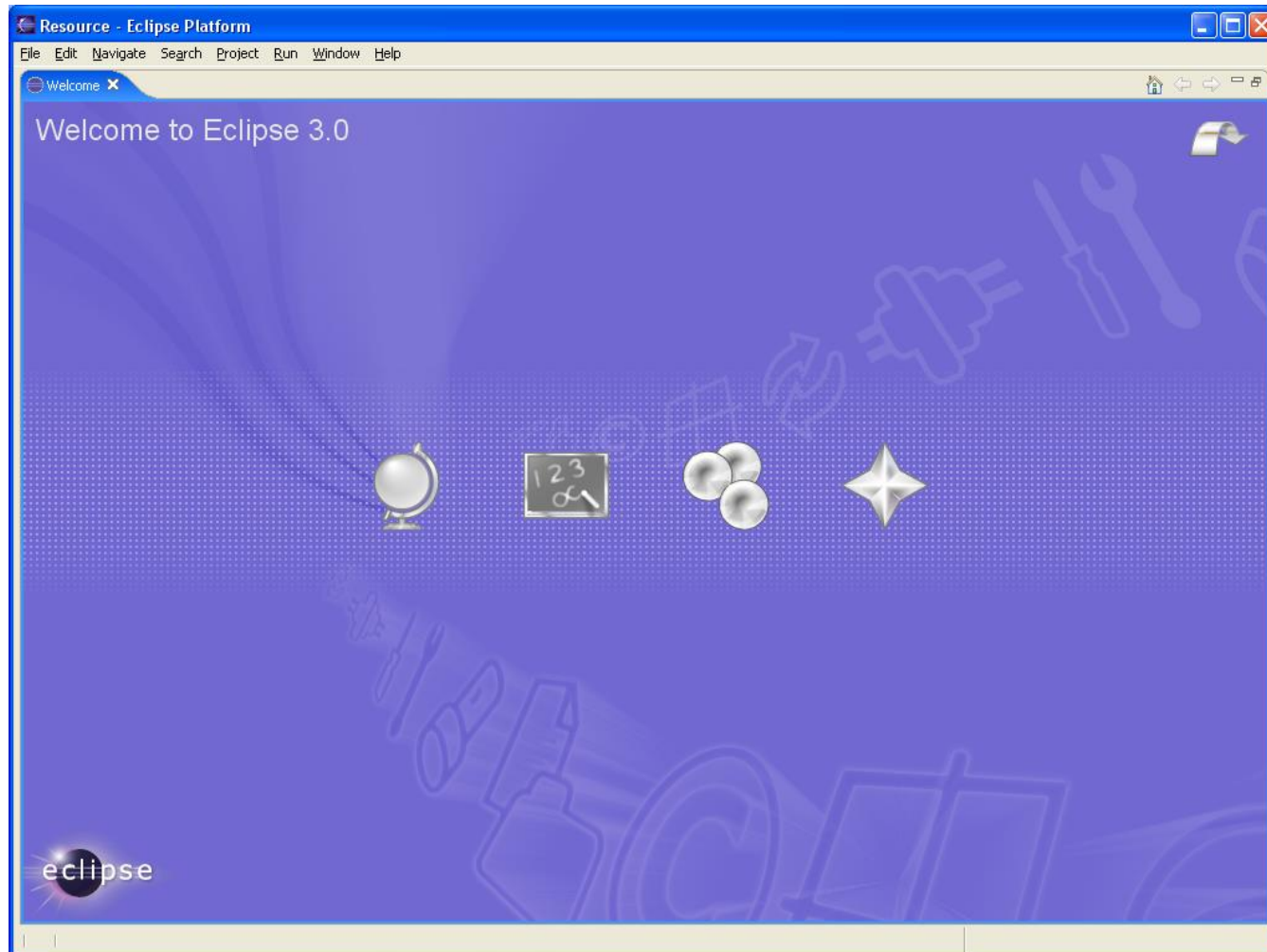
RUN ECLIPSE.EXE



SPECIFY PROJECTS DIRECTORY



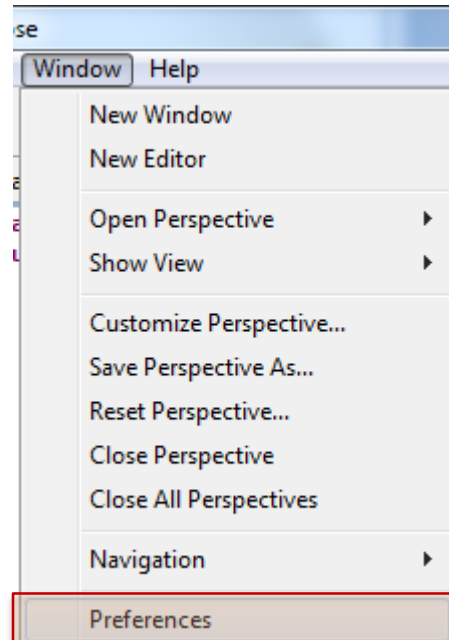
REMOVE WELCOME TAB IF IT APPEARS



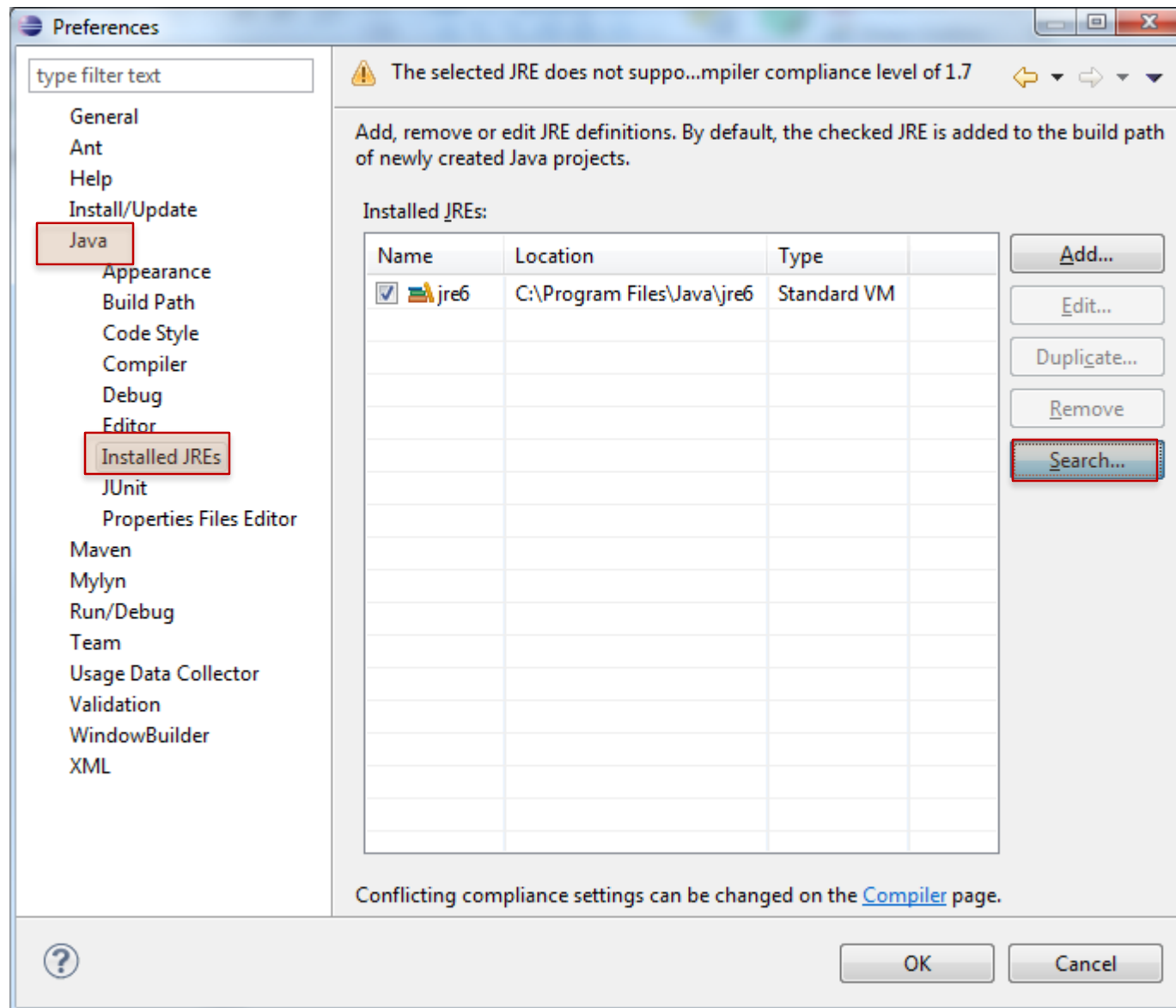
CONTENTS

- Install JDK
- Install Eclipse
- **Connect JDK to Eclipse**
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

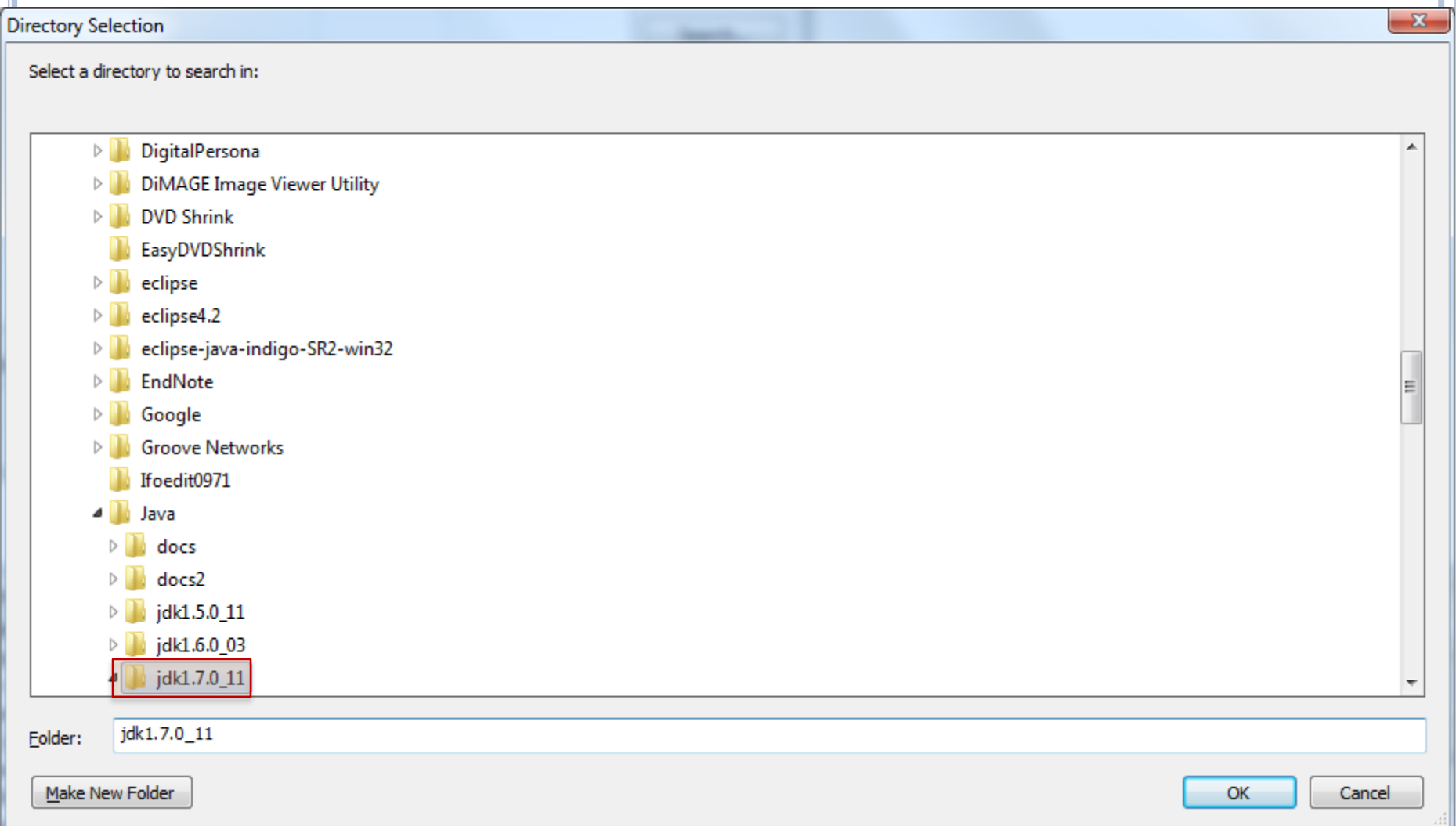
CHANGE THE JAVA VERSION: WINDOW→PREFERENCES



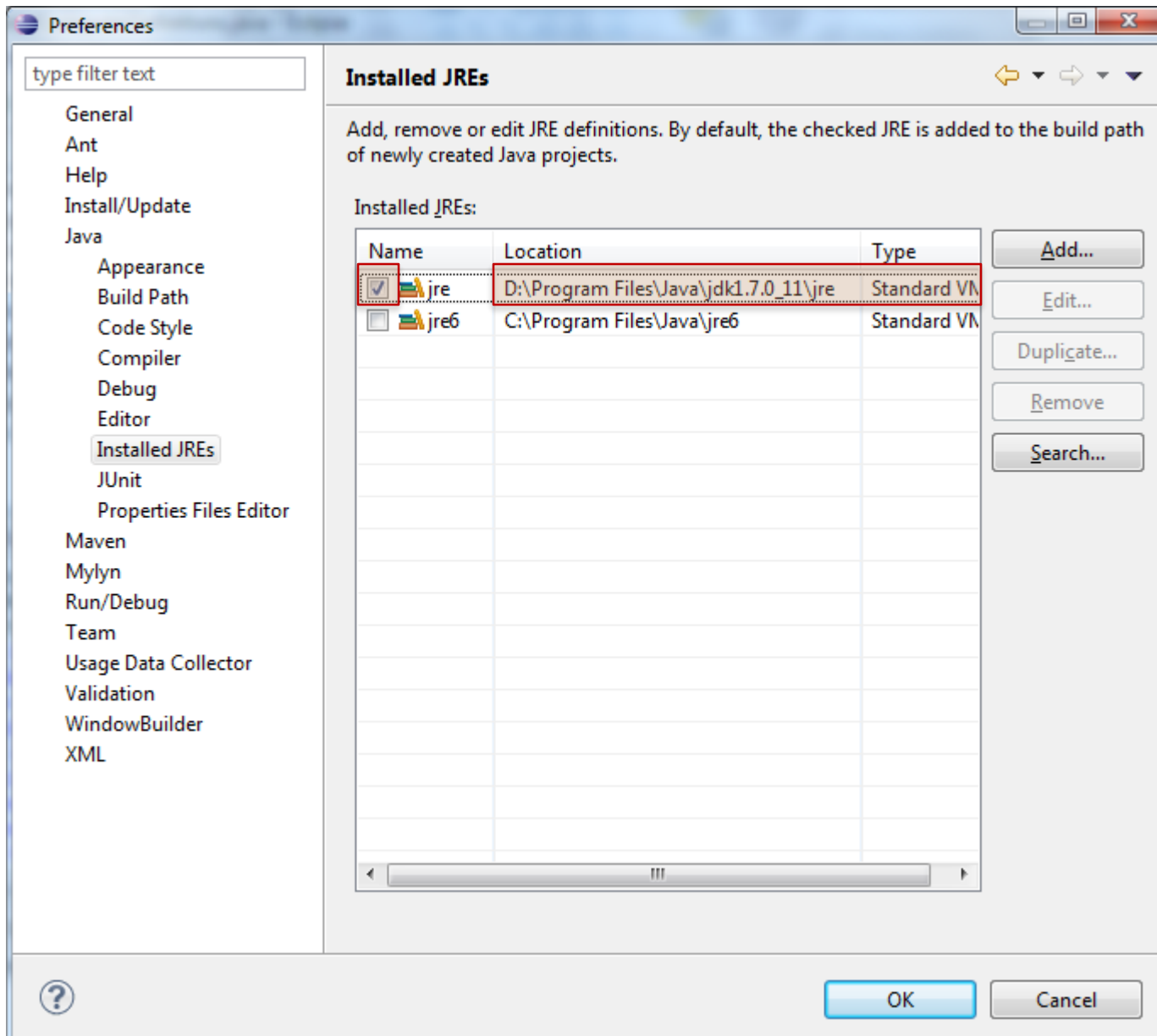
PRESS SEARCH BROWSE BUTTON



PRESS SEARCH BROWSE BUTTON TO INSTALLED JDK



JRE ADDED TO ECLIPSE, SELECT IT AND CAN EXECUTE WITHOUT ERRORS



SOME EXPLICIT INSTRUCTIONS – FOR MAC

(ASSUME JAVA 7)

You can get that for mac here:

- <https://www.oracle.com/technetwork/java/javase/downloads/jdk10-downloads-4416644.html>. Pick the Mac option and from there it should just be like installing any other application.
- Or you can go here:
<http://download.oracle.com/otn-pub/java/jdk/7u40-b43/jdk-7u40-macosx-x64.dmg>
- The problem is Eclipse won't change the compiler to use 1.7 so you will have to specify that under:
Eclipse -> Preferences -> Java -> Installed JREs
- Click "Add..." then choose "MacOS X VM"
- Next, then under directory copy/paste this:
- /Library/Java/JavaVirtualMachines/jdk1.7.0_40.jdk/Contents/Home
- Make sure it recognizes this as Java SE 7. Then name doesn't really matter.

Now go to:

Eclipse -> Preferences -> Java -> Compiler
Change the compliance level to: "1.7"

Now it probably still won't work (at least it didn't on mine)
This is because your project is still set on 1.6

So, go to:

- Project -> Properties -> Java Build Path -> Libraries
Add Library...
- JRE System Library
- "Workspace default" should be automatically there as the 1.7, but if not find the 1.7 library.
Then remove the 1.6 library
- Also you need to change the compiler being used by the project:
Project -> Properties -> Java Compiler
and make sure everything is on 1.7

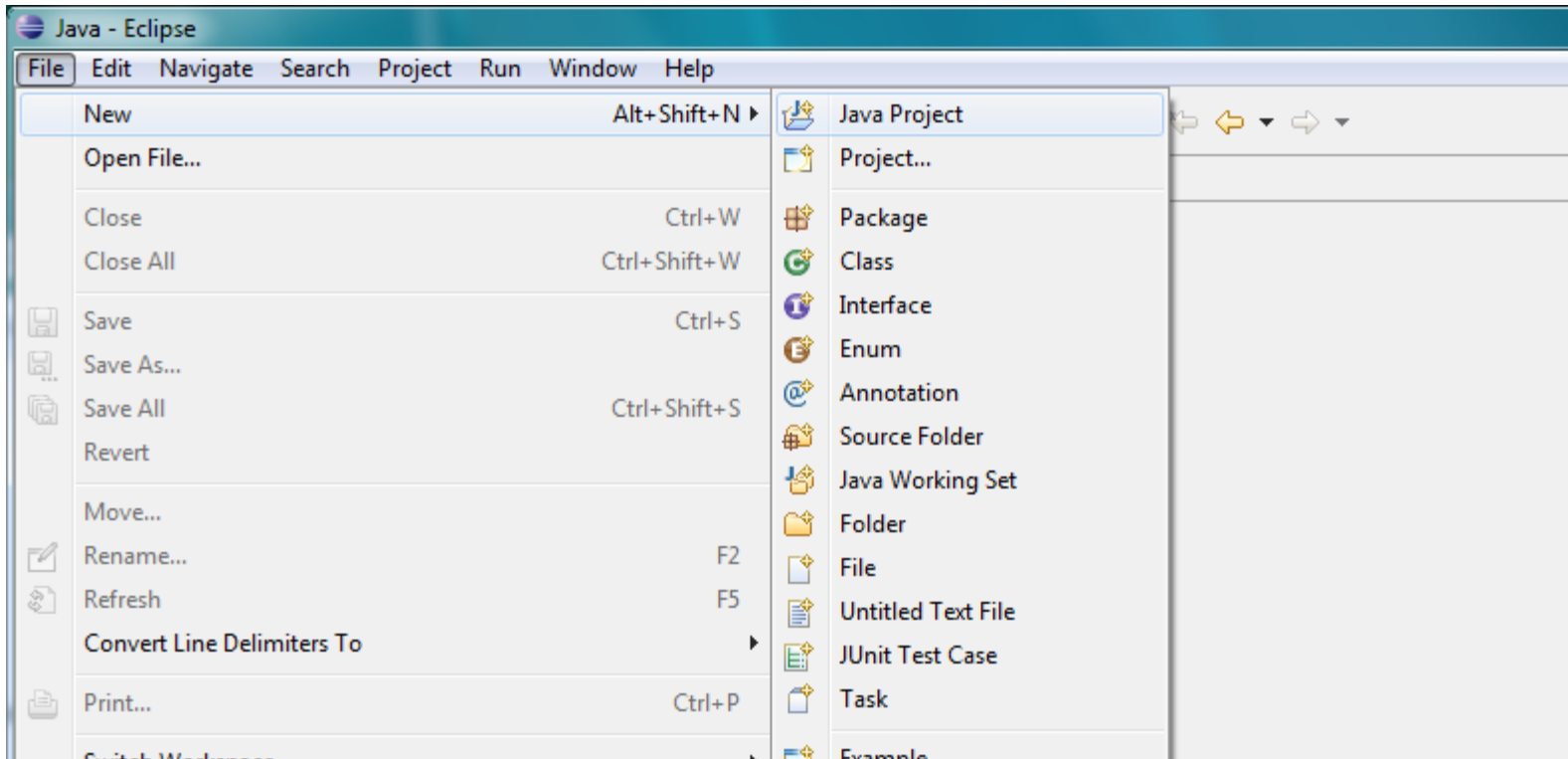
Now, if you're getting an error about the main class not being found or working.

- You may just want to create a new project and when you select the 1.7 JRE.

CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- **Create project**
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

ADD A NEW JAVA PROJECT



PROJECT NAME AND SEPARATE FOLDERS

Create a Java Project
Create a Java project in the workspace or in an external location.

Project name:

☒ Use default location
Location:

JRE

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE (currently 'jre6') [Configure JREs...](#)

Project layout

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

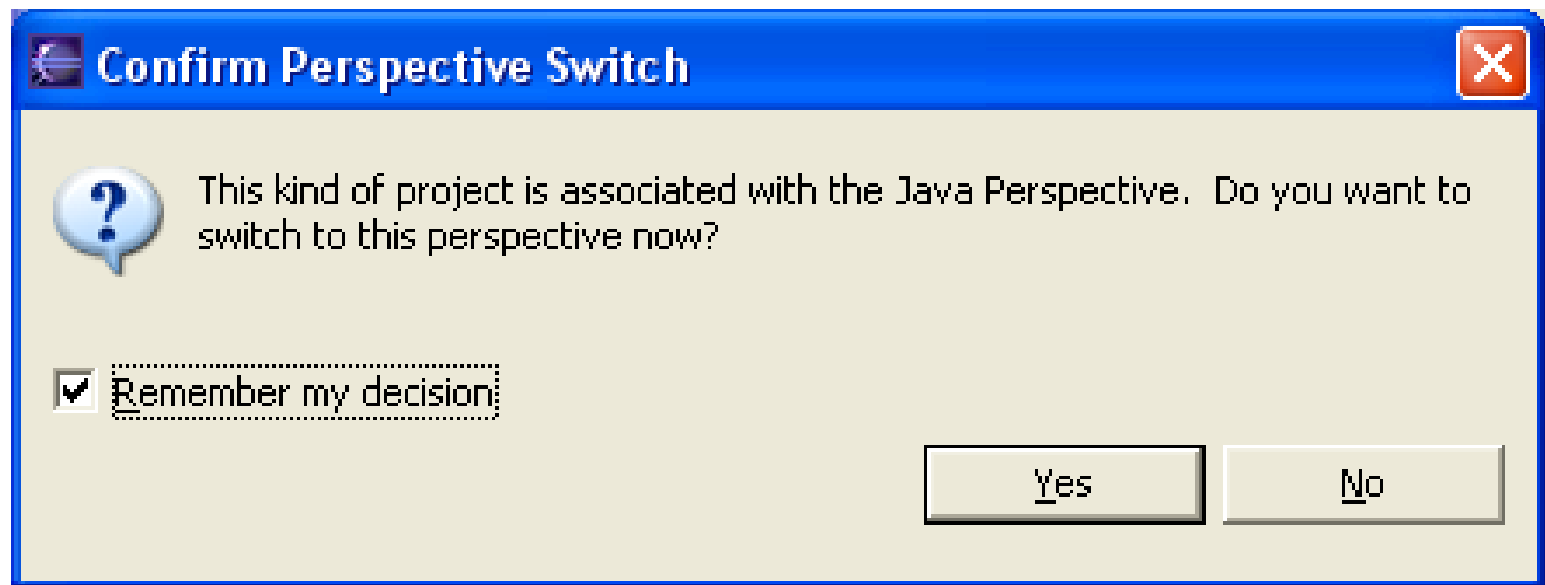
Working sets

☐ Add project to working sets
Working sets:

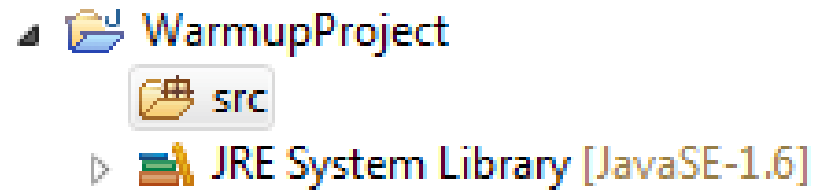
i The wizard will automatically configure the JRE and the project layout based on the existing source.

The separate folder option puts .java source files in src folder and .class files in bin folder – required for the auto grader

SWITCH TO JAVA PERSPECTIVE (MAY GET THIS MESSAGE)



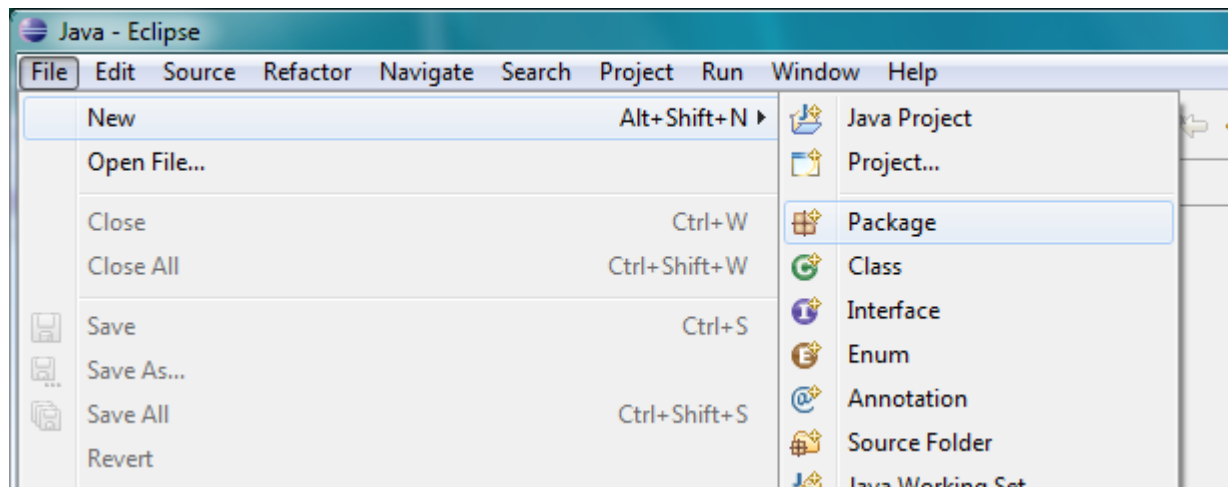
EMPTY PROJECT CREATED



CONTENTS

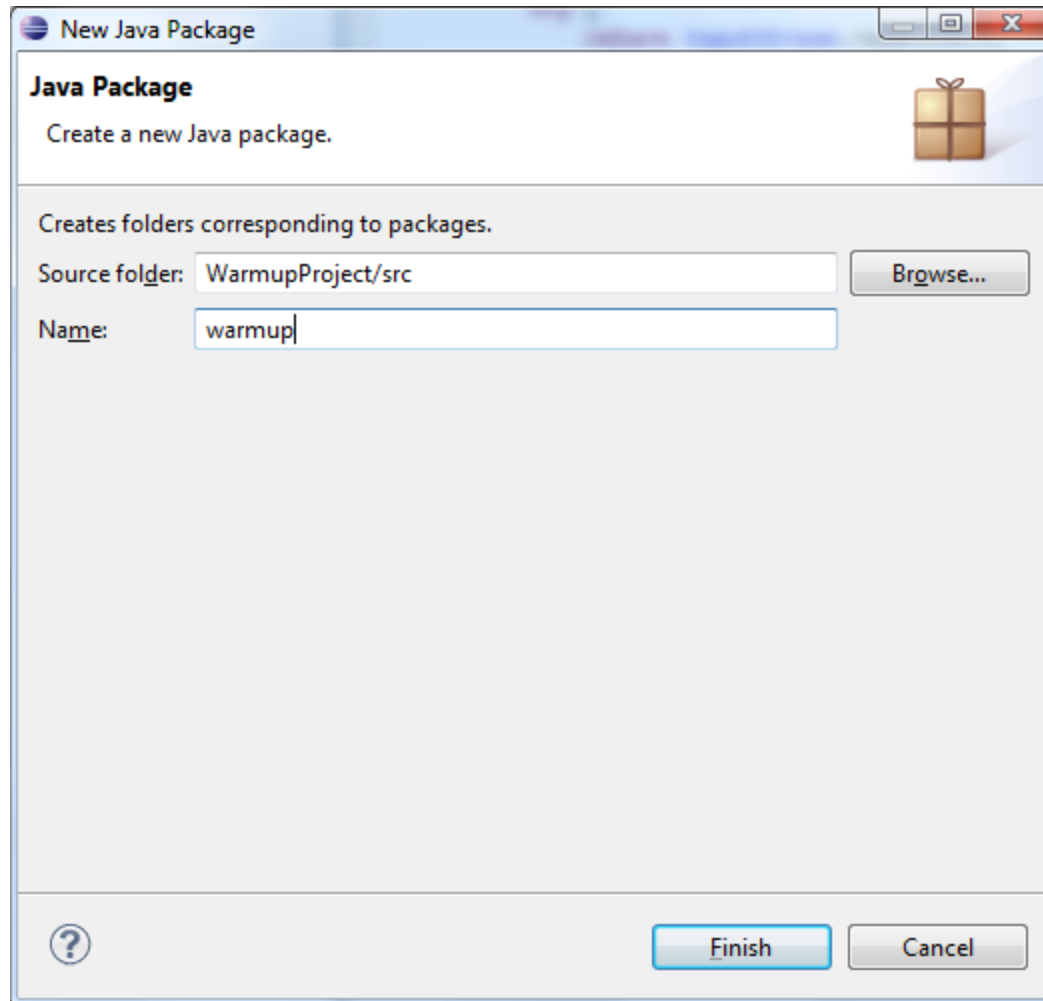
- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- **Edit run project**
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

ADD A PACKAGE TO SELECTED PROJECT



Select project and
File → Package

ENTER PACKAGE NAME



The image shows a 'New Java Package' dialog box from an IDE. It has a title bar with a blue icon and the text 'New Java Package'. The main area is titled 'Java Package' and contains the instruction 'Create a new Java package.' with a gift icon. Below this, it says 'Creates folders corresponding to packages.' There are two input fields: 'Source folder:' with the text 'WarmupProject/src' and a 'Browse...' button to its right; and 'Name:' with the text 'warmup'. At the bottom, there is a help icon (question mark), a 'Finish' button, and a 'Cancel' button.

New Java Package

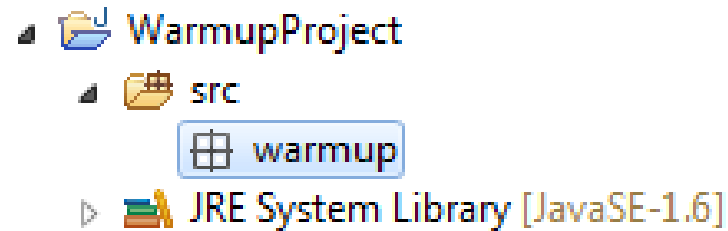
Java Package
Create a new Java package.

Creates folders corresponding to packages.

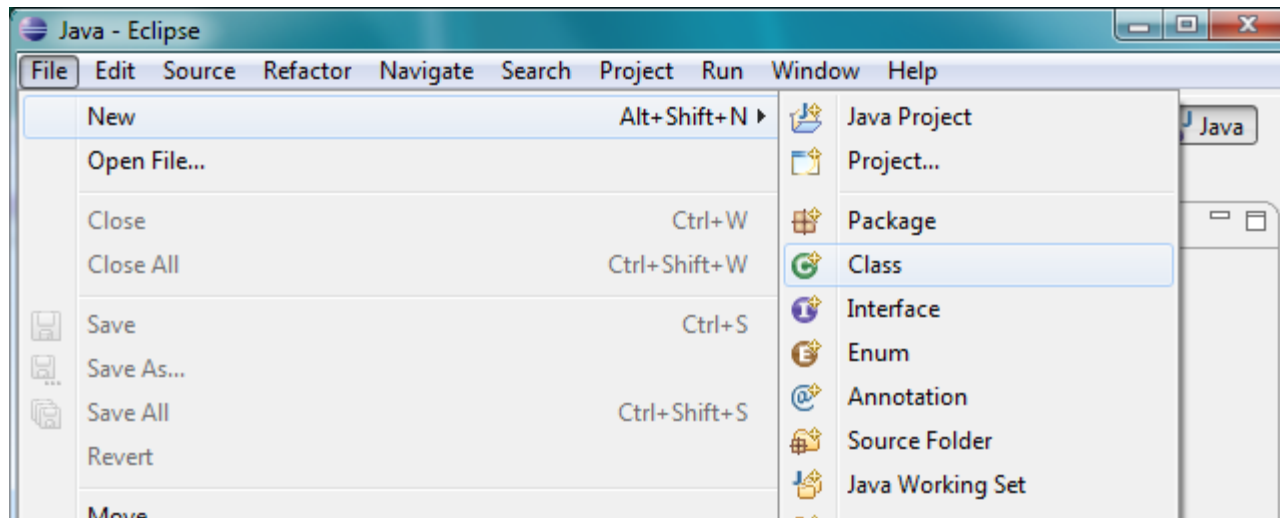
Source folder: WarmupProject/src

Name: warmup

EMPTY PACKAGE IN NON EMPTY PROJECT



ADD A CLASS



Select project and
File → Class

NAME THE CLASS

New Java Class

Create a new Java class.

Source folder: WarmupProject/src **Browse...**

Package: warmup **Browse...**

☐ Enclosing type: **Browse...**

Name: AGreeter

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object **Browse...**

Interfaces: **Add...**
Remove

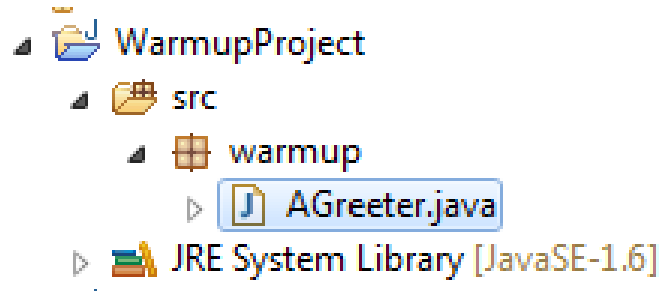
Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Finish **Cancel**

EMPTY CLASS IN NON EMPTY PACKAGE



```
AGreeter.java X
package warmup;

public class AGreeter {

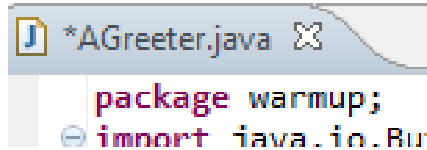
}
```

EDIT THE FILE (COPY AND PASTE CODE)

```
package warmup;
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class AGreeter {
    static BufferedReader inputStream = new BufferedReader(new InputStreamReader(System.in));
    public static void main(String[] args) {
        System.out.println("Hello World");
        if (args.length > 0) {
            System.out.println(args[0]);
        }
        try {
            System.out.println (inputStream.readLine());
        } catch (Exception e) {
            System.out.println("Could not read line");
            e.printStackTrace();
        }
    }
}
```

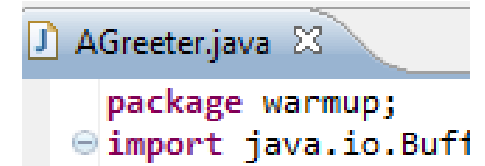
Copy and paste code from:

SAVE (AND COMPILE) THE FILE



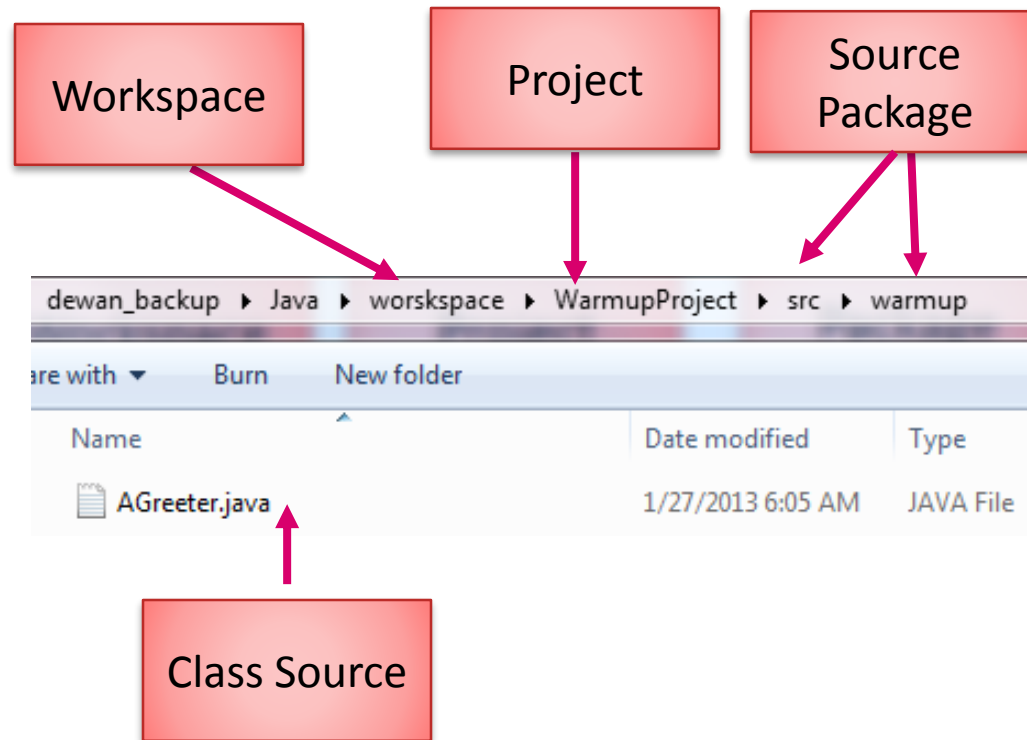
```
*AGreeter.java X  
package warmup;  
import java.io.Buf
```

File → Save or CTRL+S

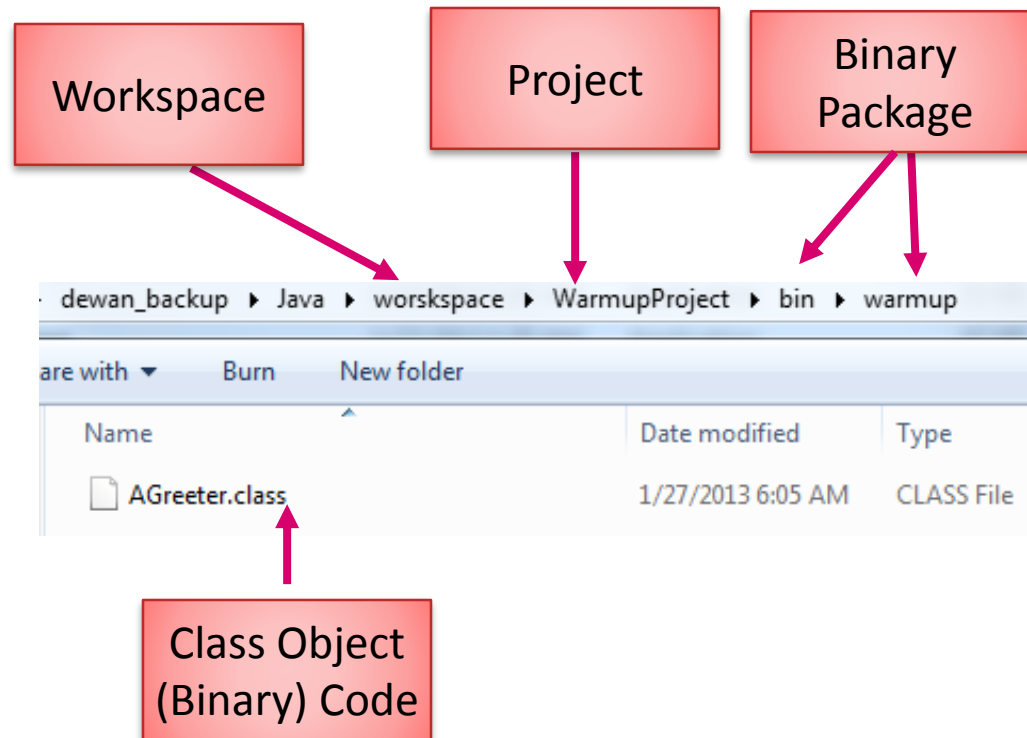


```
AGreeter.java X  
package warmup;  
import java.io.Buf
```

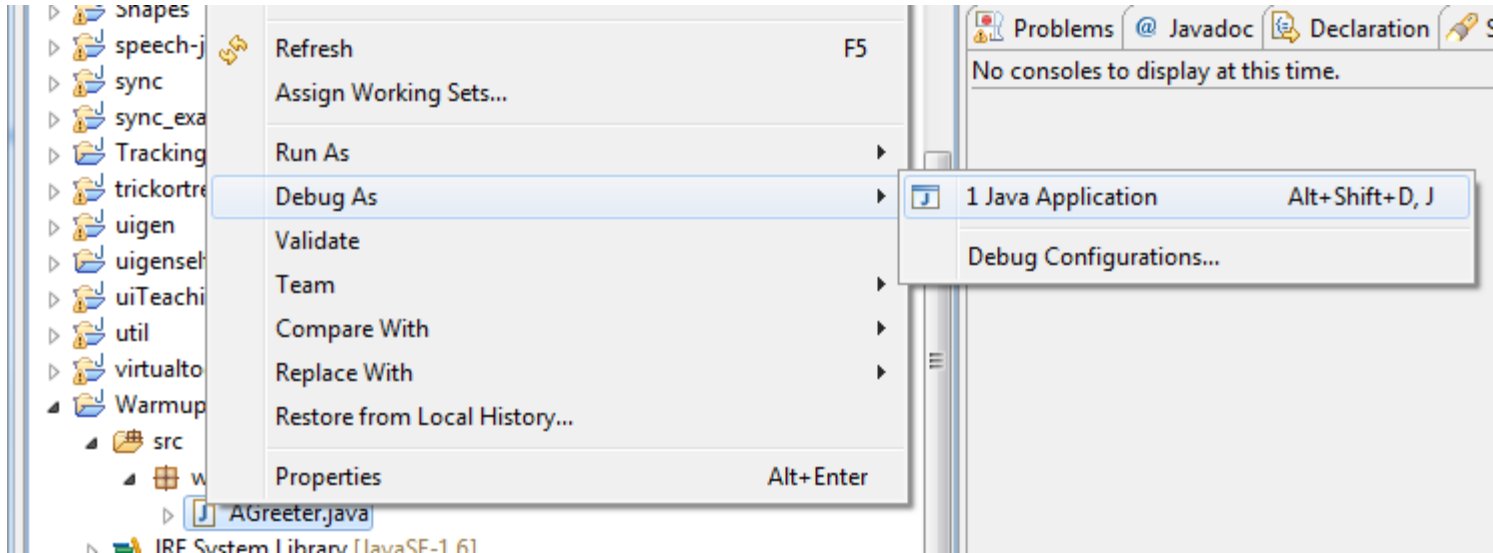
WORKSPACE, PROJECT, SOURCE PACKAGE → FOLDER



WORKSPACE, PROJECT, BINARY PACKAGE → FOLDER



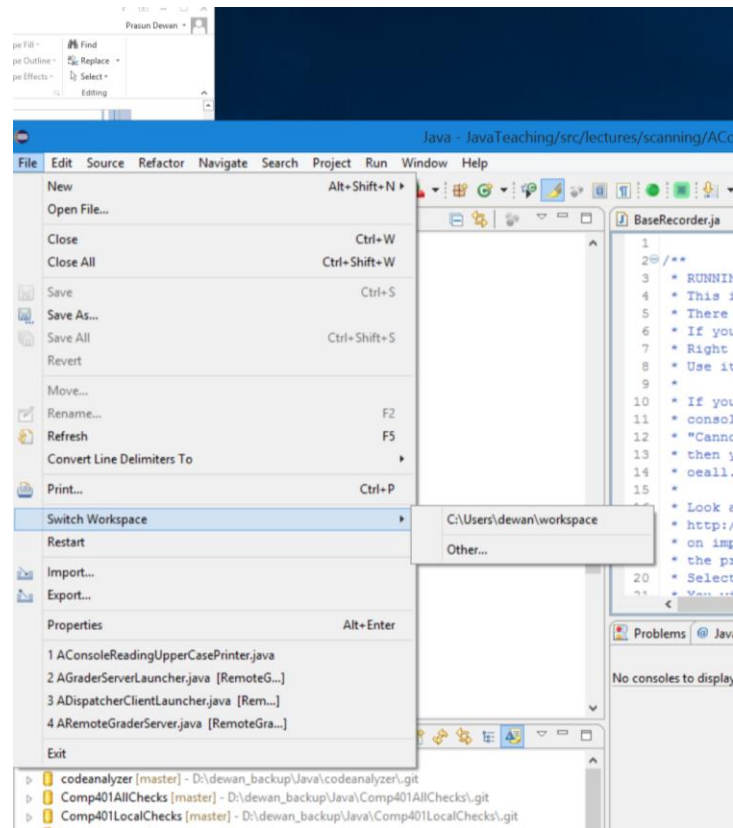
EXECUTE THE MAIN METHOD ON SELECTED CLASS



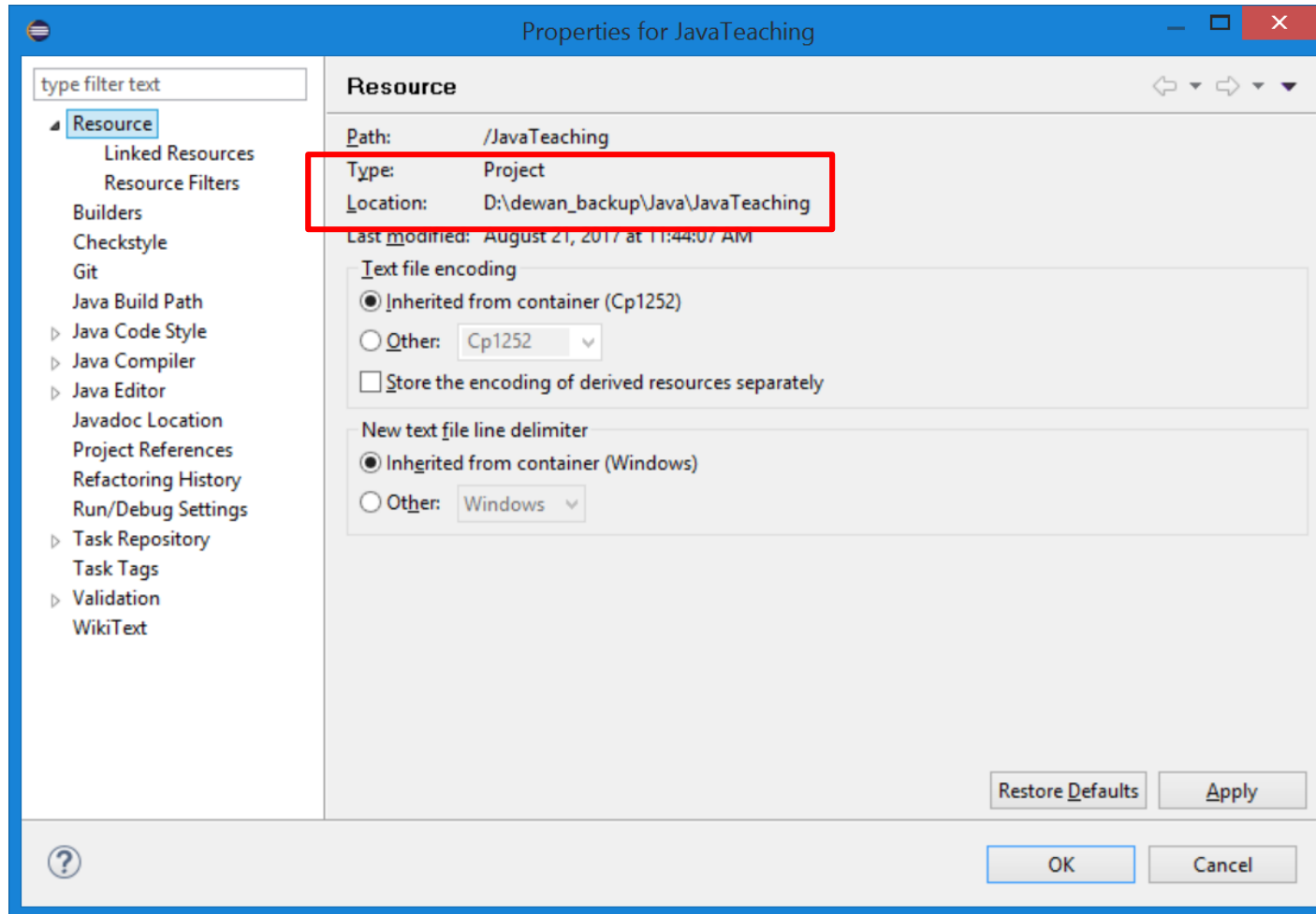
Right Menu → Debug As → Java Application

Can also select Run but Debug is more defensive

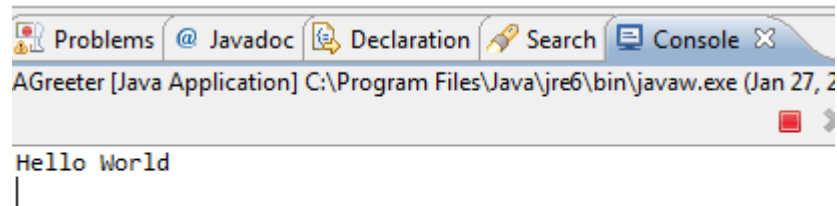
FIND OR SWITCH WORKSPACE FOLDER: FILE→SWITCH WORKSPACE



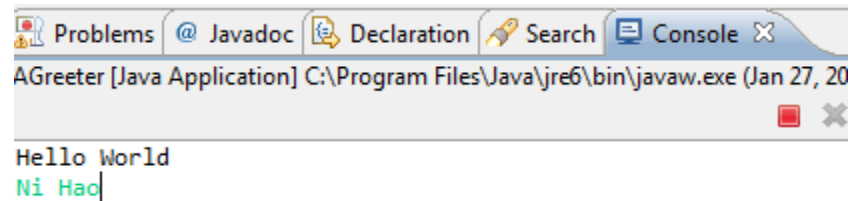
FIND PROJECT FOLDER: SELECT PROJECT, RIGHT CLICK→PROPERTIES→RESOURCE



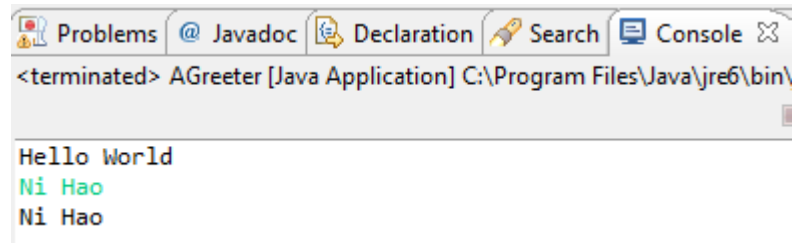
VIEW OUTPUT IN CONSOLE WINDOW



PROVIDE INPUT IN CONSOLE WINDOW



THE OUTPUT AFTER ENTER



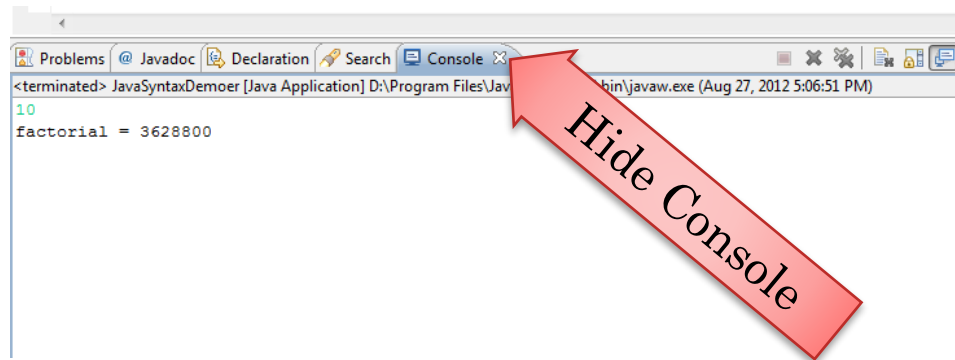
The screenshot shows a console window from an IDE. The title bar includes tabs for 'Problems', 'Javadoc', 'Declaration', 'Search', and 'Console'. The console text reads: '<terminated> AGreeter [Java Application] C:\Program Files\Java\jre6\bin\' followed by three lines of output: 'Hello World', 'Ni Hao' (with 'Ni' in green), and 'Ni Hao'.

```
<terminated> AGreeter [Java Application] C:\Program Files\Java\jre6\bin\  
Hello World  
Ni Hao  
Ni Hao
```








CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- **Restore window**
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

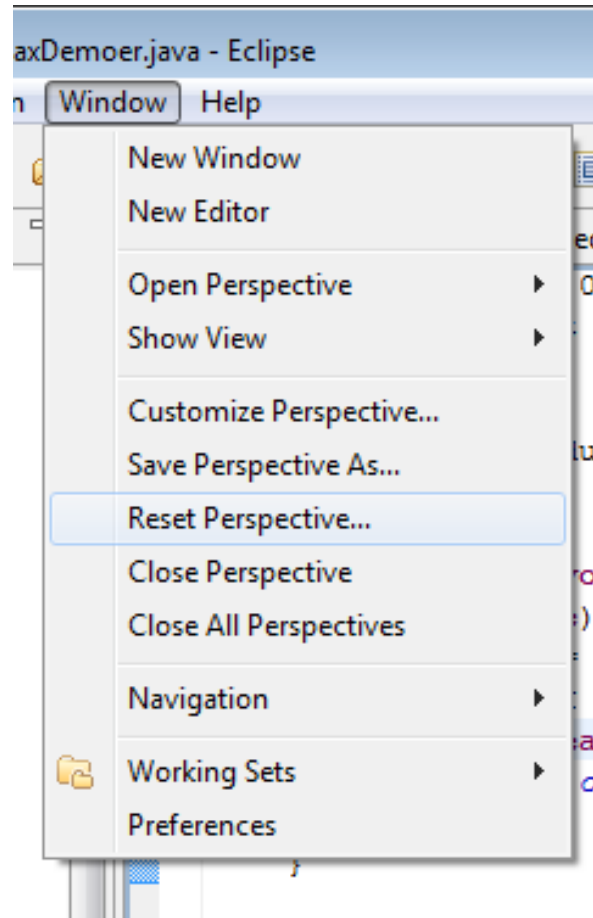
CLOSING A WINDOW PERHAPS ACCIDENTALLY



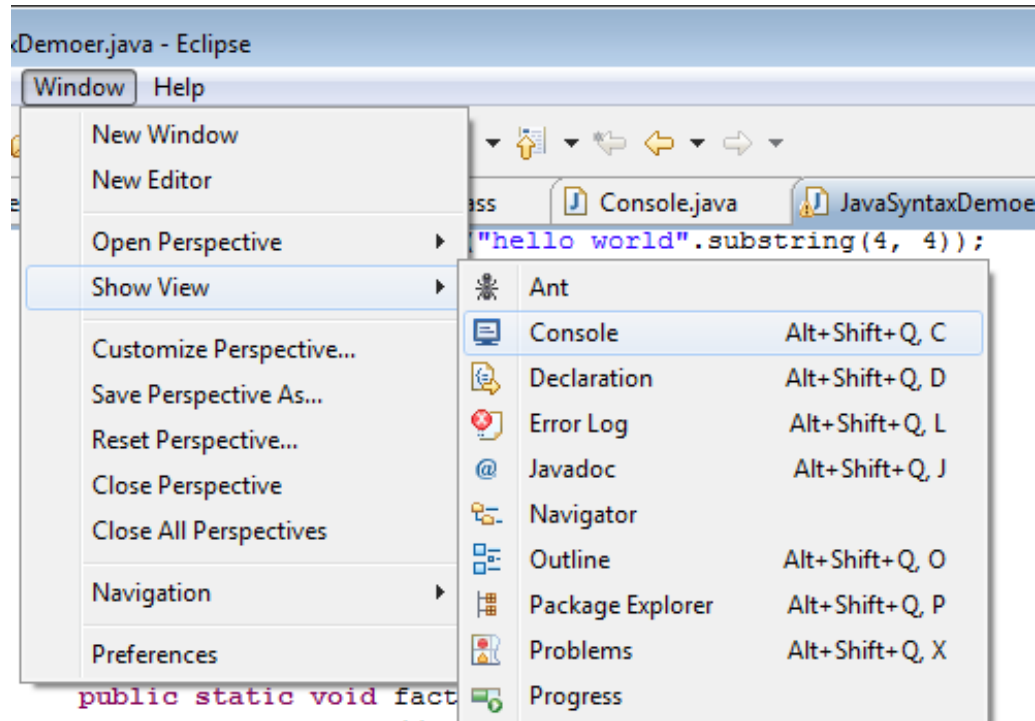
VANISHED CONSOLE

 Problems   Javadoc  Declaration  Search	
4 errors, 3,673 warnings, 0 others (Filter matched 104 of 3677 items)	
Description	Resource
 Errors (4 items)	
 Warnings (100 of 3673 items)	

WINDOW → RESET PERSPECTIVE



WINDOW → SHOW VIEW



Can selectively remove and add sub-windows/views

CONTENTS

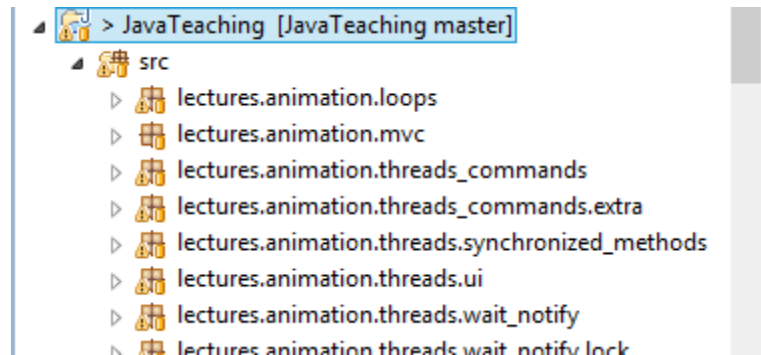
- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- **Search project for strings**
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

FILE SEARCH IN A PROJECT (USEFUL IN JAVA TEACHING AND YOUR PROJECTS)

Suppose you want to search the entire project for some string such as some text in a Sakai question: such as “println(c) displays”

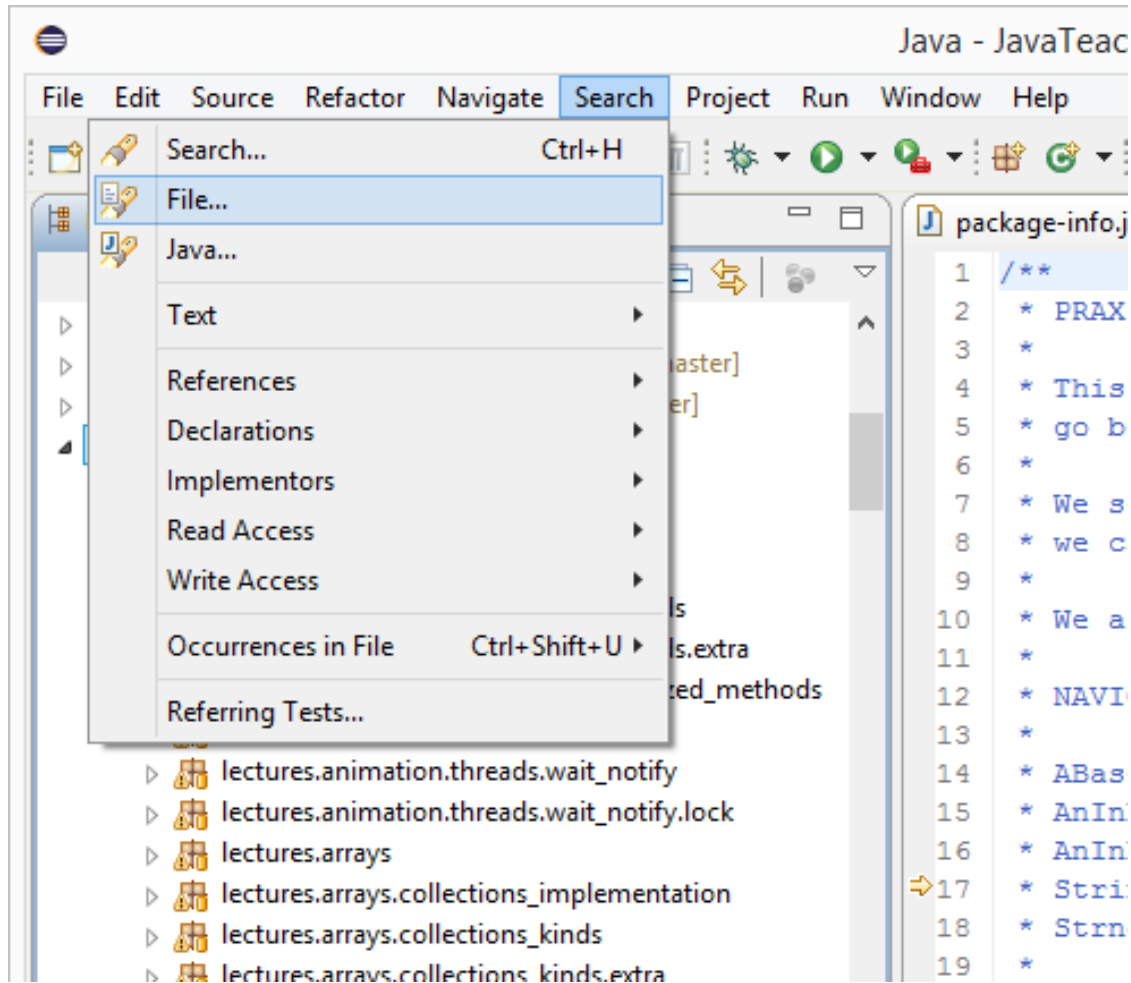
SELECT PROJECT

Suppose you want to search the entire project for some string such as some text in a Sakai question: such as “println(c) displays”



Select project (not the src folder)

FILE → SEARCH



SEARCH DIALOG

The image shows the Eclipse Search dialog box. It has a title bar with standard window controls. Below the title bar is a tabbed interface with five tabs: 'File Search' (selected), 'Task Search', 'Git Search', 'Java Search', and 'Plug-in Search'. The 'File Search' tab is active, showing the following options:

- Containing text:** A text field containing 'println(c) displays'. A blue callout box labeled 'Search string' points to this field.
- File name patterns (separated by comma):** A dropdown menu showing '*.java' and a 'Choose...' button.
- Scope:** Three radio buttons: 'Workspace' (unselected), 'Selected resources' (unselected), and 'Enclosing projects' (selected). A blue callout box labeled 'Scope of search' points to the 'Enclosing projects' radio button.
- Working set:** A text field and a 'Choose...' button.

At the bottom of the dialog, there are four buttons: a help button (question mark icon), 'Customize...', 'Replace...', and 'Search'. A blue callout box labeled 'Command' points to the 'Search' button.

Additional options on the right side of the 'Containing text' section include:

- ☒ Case sensitive
- ☐ Regular expression
- ☐ Whole word

Legend for file name patterns: (* = any string, ? = any character, \ = escape for literals: * ? \)

Legend for scope: (* = any string, ? = any character, !x = excluding x)

SEARCH RESULTS

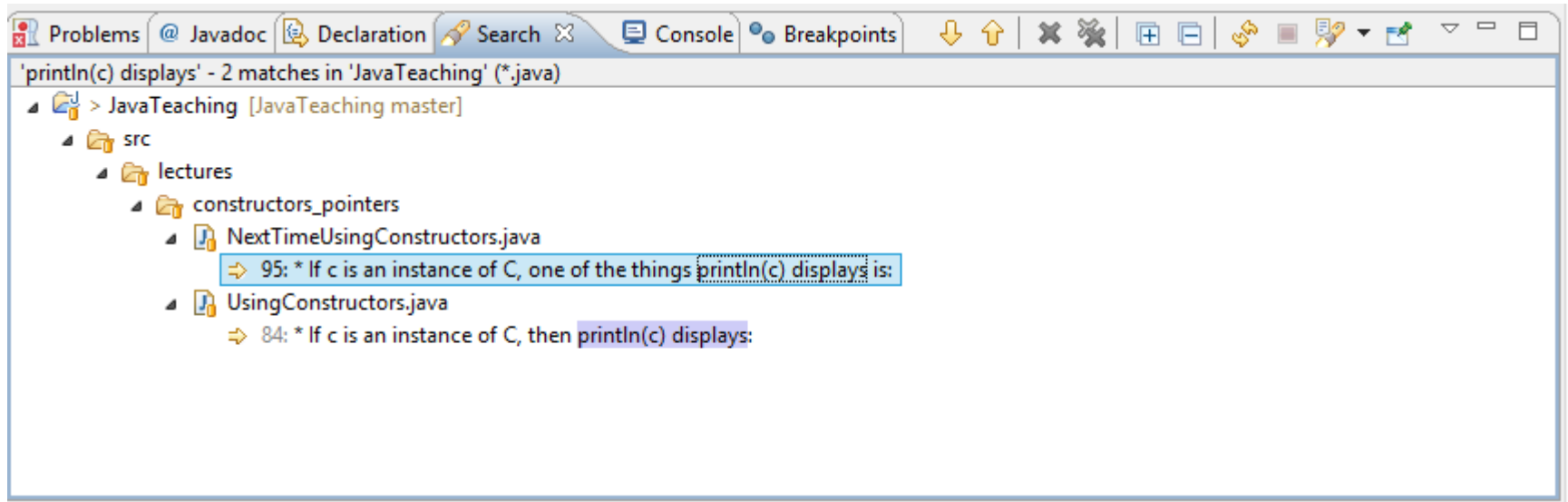
The screenshot shows the Eclipse IDE interface with the following components:

- Package Explorer:** Displays the project structure, including the `JavaTeaching` project and its sub-packages like `lectures.animation.loops`, `lectures.animation.mvc`, etc.
- Source Editor:** Shows the `package-info.java` file with the following content:

```
1 /**
2  * PRAXIS INFO
3  *
4  * This praxis teaches inheritance using the example of collections that
5  * go beyond fixed-length arrays.
6  *
7  * We see three such collections- histories, databases, and sets- and see how
8  * we can reuse (share) code among them using inheritance.
9  *
10 * We also see factory methods that choose between different histories.
11 *
12 * NAVIGATION INFO
13 *
14 * ABaseStringHistory
15 * AnInheritingStringDatabase
16 * AnInheritingStringSet
17 * StringHistoryFactory
18 * StringHistoryReverser
19 *
20 */
21 package lectures.inheritance;
```
- Search View:** Shows the search results for the query `'println(c) displays' - 2 matches in 'JavaTeaching' (*.java)`. The results are listed under the `JavaTeaching` project and its sub-packages, including `lectures.constructors_pointers` and `lectures.using_constructors`.
- Console:** Shows the output of the search, including the text: `95: * If c is an instance of C, one of the things println(c) displays:` and `84: * If c is an instance of C, then println(c) displays:`.

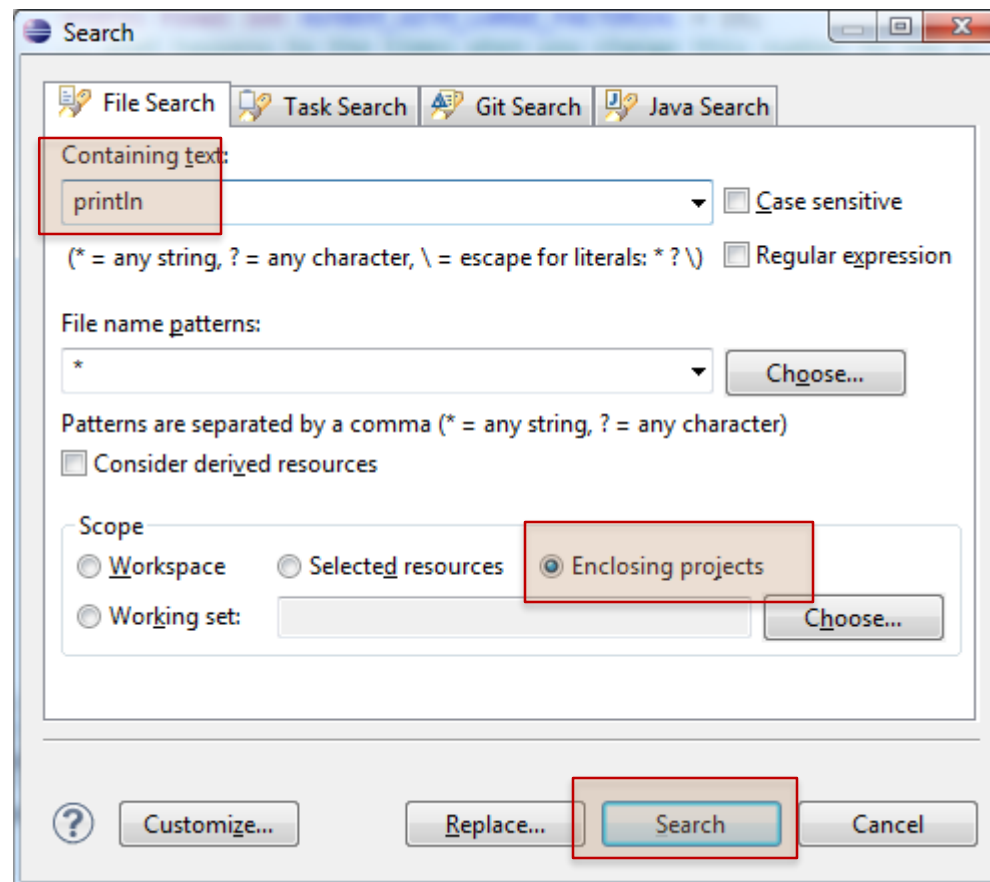
A blue callout box on the right side of the image contains the text: "Search view shown, console in another tab".

SEARCH RESULTS ZOOMED

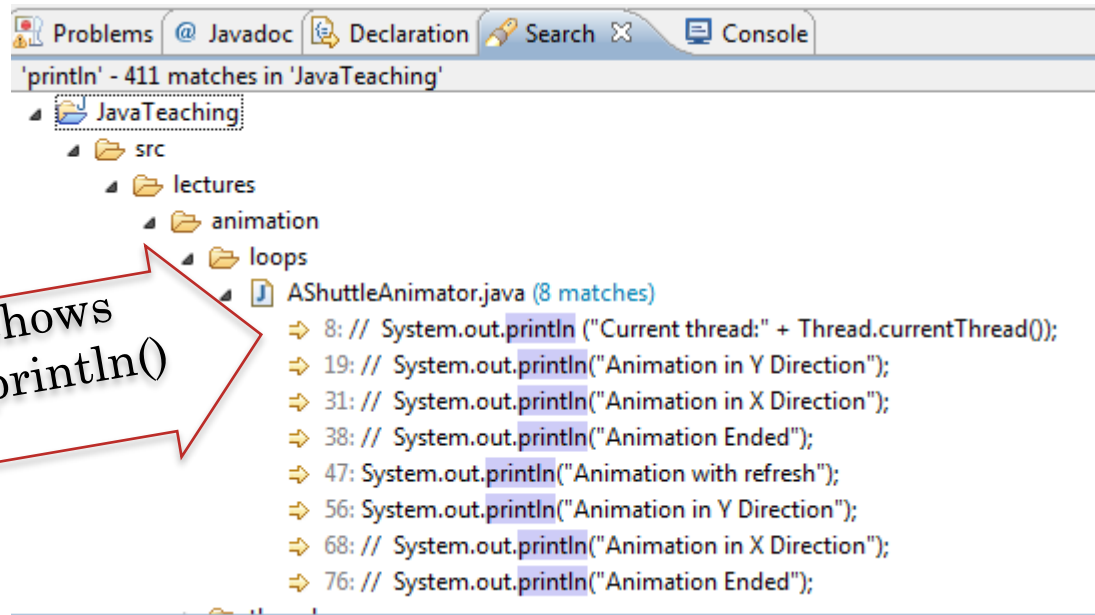


Look at all the tabs
availalale here

MULTI FILE TEXT SEARCH FOR PRINTLN



SEARCH RESULTS



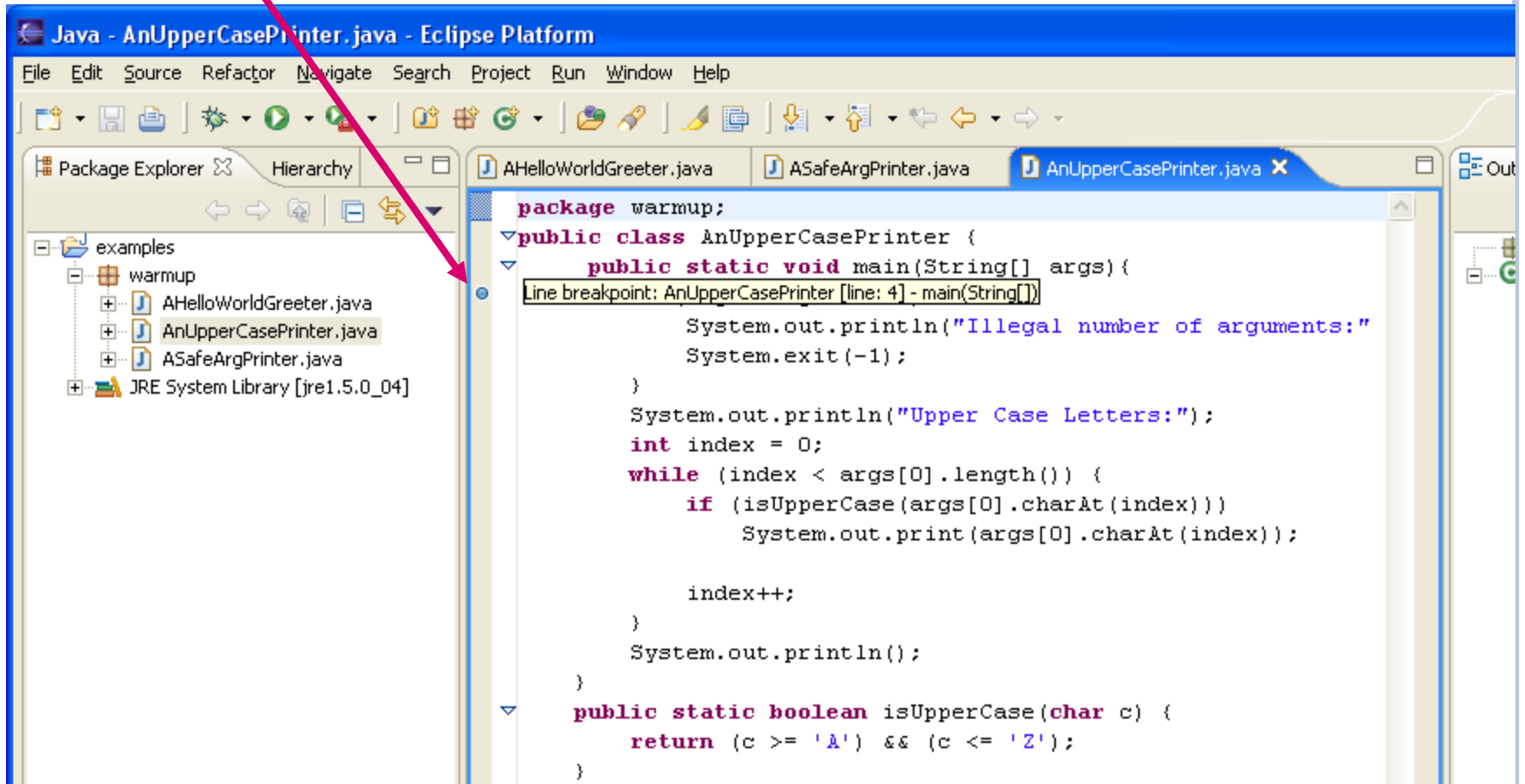
Use CTR+SHIFT+G (shown earlier) to find active uses of the method

CONTENTS

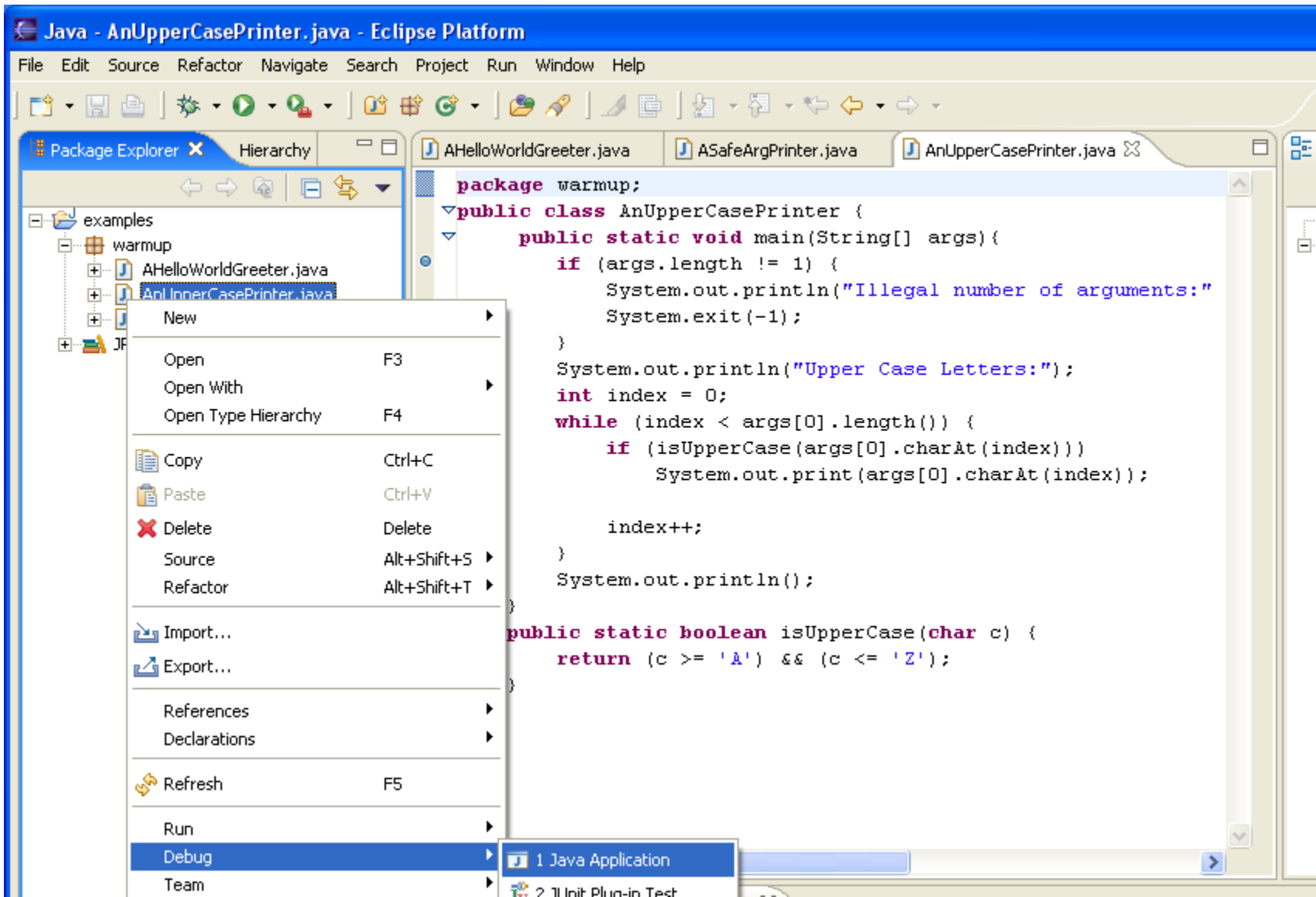
- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- **Debugging**
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

SETTING A BREAK POINT

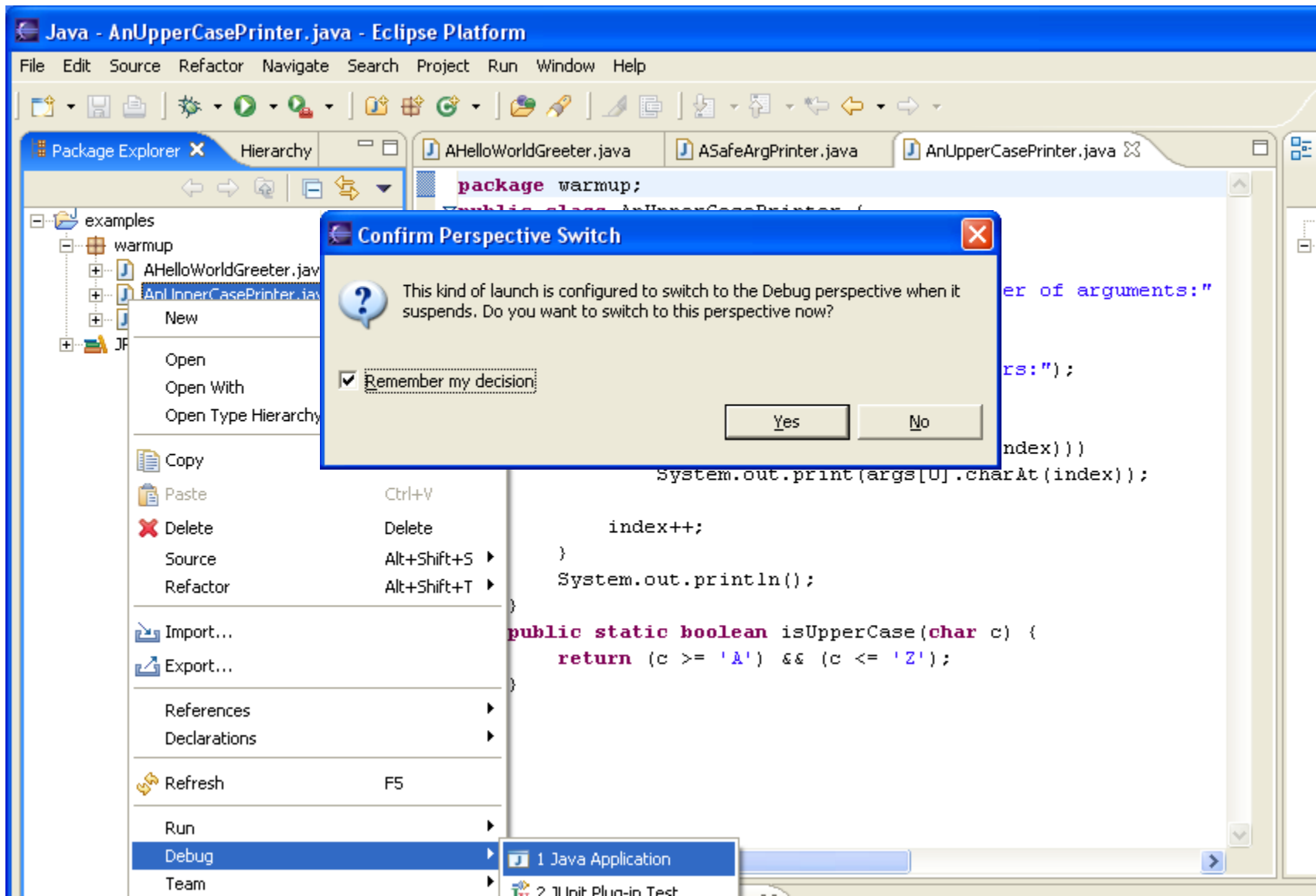
Double click where you want program to stop



DEBUG-RUN



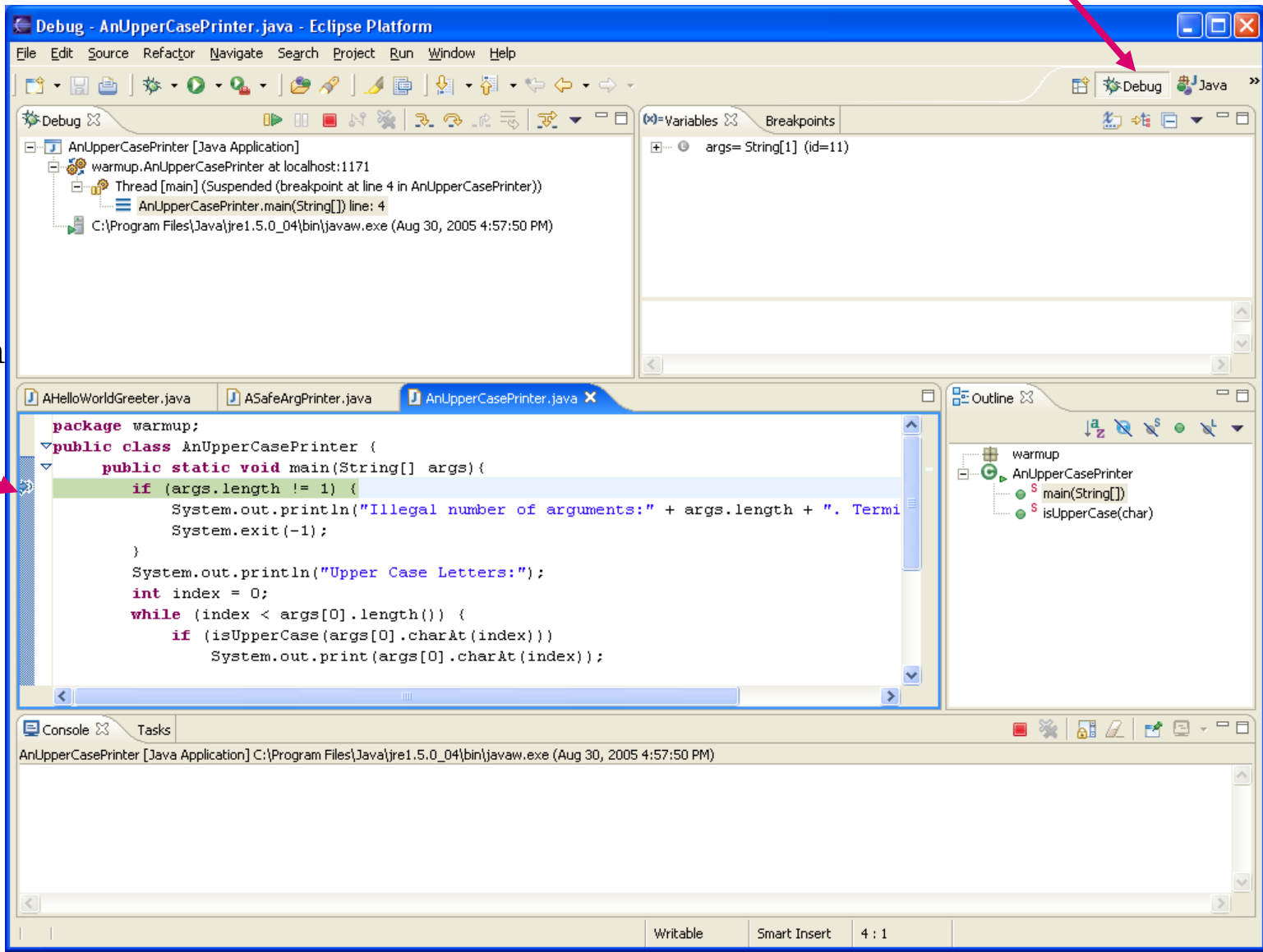
STARTING THE DEBUGGER



STOPPING AT BREAKPOINT

Debug Perspective

Next
statement
to be
executed



EXAMINING MAIN PARAMETERS

The screenshot displays the Eclipse IDE interface during a debug session. The top toolbar includes icons for file operations, running, and debugging. The left sidebar shows the project structure with 'AnUpperCasePrinter [Java Application]' expanded, revealing a 'Thread [main]' and the current execution point at 'AnUpperCasePrinter.main(String[]) line: 4'. The bottom-left pane shows the source code of 'AnUpperCasePrinter.java', with the 'main' method highlighted. The bottom-right pane shows the 'Outline' view with the 'main' method selected. The top-right pane shows the 'Variables' window, which displays the current state of the program's variables. A red arrow points to the 'args' variable, which is a String array of length 1, containing the value 'John F. Kennedy'. The console at the bottom shows the output of the program, which is 'Upper Case Letters:'.

Debug - AnUpperCasePrinter.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Debug

AnUpperCasePrinter [Java Application]

- warmup.AnUpperCasePrinter at localhost:1171
 - Thread [main] (Suspended (breakpoint at line 4 in AnUpperCasePrinter))
 - AnUpperCasePrinter.main(String[]) line: 4
- C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Aug 30, 2005 4:57:50 PM)

Variables Breakpoints

args= String[1] (id=11)

[0]= "John F. Kennedy"

Main Parameters

[John F. Kennedy]

Outline

```
package warmup;
public class AnUpperCasePrinter {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Illegal number of arguments:" + args.length + ". Terminate.");
            System.exit(-1);
        }
        System.out.println("Upper Case Letters:");
        int index = 0;
        while (index < args[0].length()) {
            if (isUpperCase(args[0].charAt(index)))
                System.out.print(args[0].charAt(index));
        }
    }
}
```

Console Tasks

AnUpperCasePrinter [Java Application] C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Aug 30, 2005 4:57:50 PM)

STEPPING TO NEXT STATEMENT

The screenshot shows the Eclipse IDE interface during a Java debug session. The title bar reads "Debug - AnUpperCasePrinter.java - Eclipse Platform". The menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The Run menu is open, displaying various debugging actions with their corresponding keyboard shortcuts. The "Step Over" option is highlighted in blue. In the background, the "Breakpoints" view shows a breakpoint at line 4 of AnUpperCasePrinter.java, and the "Outline" view shows the project structure with the current class selected.

Debug - AnUpperCasePrinter.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Toggle Line Breakpoint Ctrl+Shift+B

Toggle Method Breakpoint

Toggle Watchpoint

Skip All Breakpoints

Add Java Exception Breakpoint...

Add Class Load Breakpoint...

Resume F8

Suspend

Terminate

Step Into F5

Step Over F6

Step Return F7

Run to Line Ctrl+R

Use Step Filters Shift+F5

Run Last Launched Ctrl+F11

Debug Last Launched F11

Run History

Run As

Run...

Debug History

Debug As

Debug...

Insert Ctrl+Shift+I

Breakpoints

String[1] (id=11)

] = "John F. Kennedy"

ennedy]

Outline

warmup

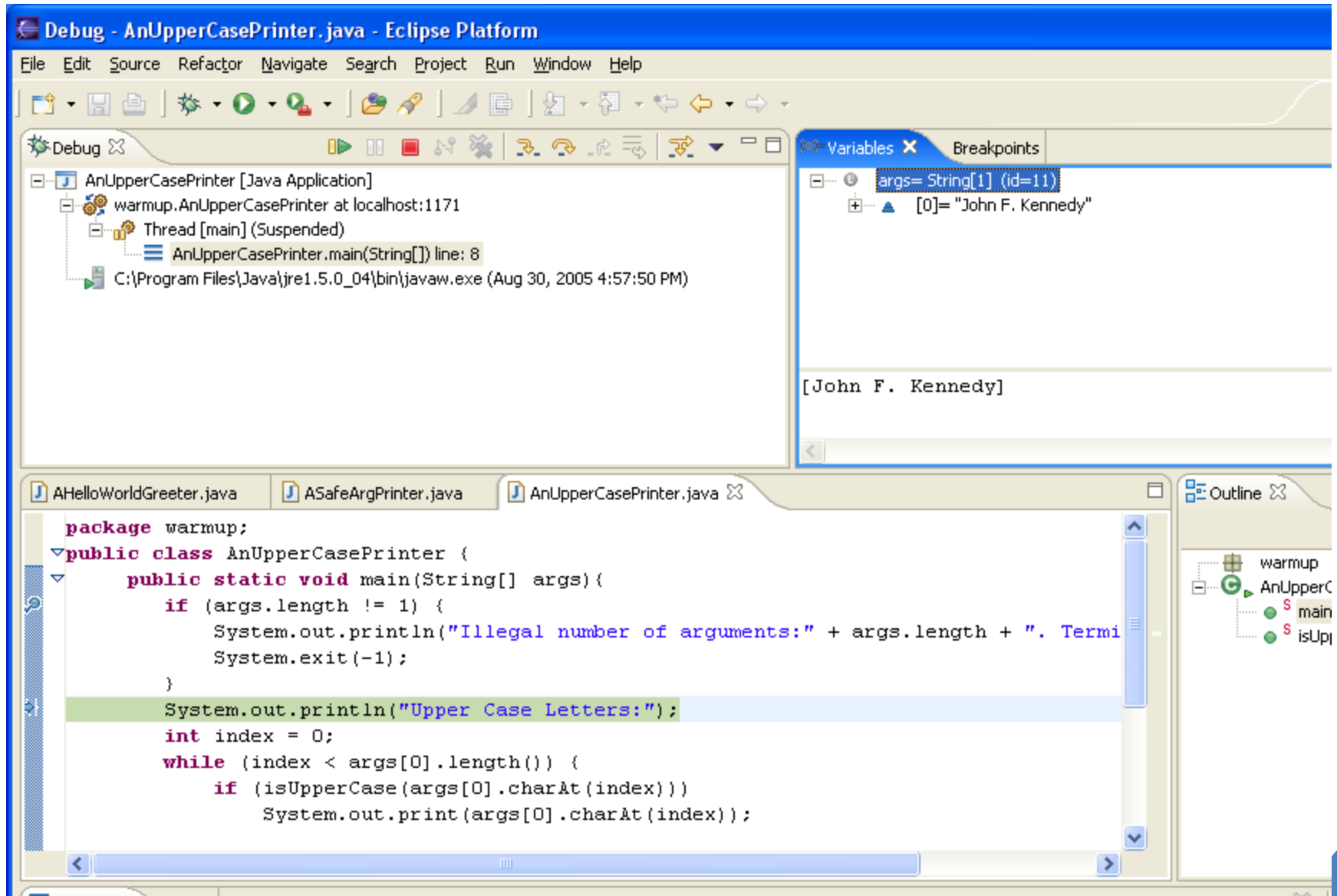
AnUppe

S ma

S isU

```
package warmup;
public class AnUpperCasePrinter {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Invalid number of arguments. Terminating.");
            System.exit(-1);
        }
        System.out.println("Upper case printer");
        int index = 0;
        while (index < args[0].length()) {
            if (Character.isUpperCase(args[0].charAt(index))) {
                System.out.print(args[0].charAt(index) + " ");
            } else {
                System.out.print(Character.toUpperCase(args[0].charAt(index)) + " ");
            }
            index++;
        }
        System.out.println();
    }
}
```

NEXT STATEMENT



Debug - AnUpperCasePrinter.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Debug

AnUpperCasePrinter [Java Application]

- warmup.AnUpperCasePrinter at localhost:1171
 - Thread [main] (Suspended)
 - AnUpperCasePrinter.main(String[]) line: 8
 - C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Aug 30, 2005 4:57:50 PM)

Variables

args= String[1] (id=11)

- [0]= "John F. Kennedy"

[John F. Kennedy]

Outline

- warmup
 - AnUpperC
 - main
 - isUp

```
package warmup;
public class AnUpperCasePrinter {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Illegal number of arguments:" + args.length + ". Terminating.");
            System.exit(-1);
        }
        System.out.println("Upper Case Letters:");
        int index = 0;
        while (index < args[0].length()) {
            if (isUpperCase(args[0].charAt(index)))
                System.out.print(args[0].charAt(index));
        }
    }
}
```

NEW OUTPUT AND LOCAL VARIABLE

Debug - AnUpperCasePrinter.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Debug

AnUpperCasePrinter [Java Application]

- warmup.AnUpperCasePrinter at localhost:1171
 - Thread [main] (Suspended)
 - AnUpperCasePrinter.main(String[]) line: 11
 - C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Aug 30, 2005 4:57:50 PM)

Variables

- args= String[1] (id=11)
 - [0]= "John F. Kennedy"
 - index= 0

Index gets initialized

[John F. Kennedy]

Outline

- warmup
 - AnUpperCasePrinter
 - main(String)
 - isUpperCas

```
package warmup;
public class AnUpperCasePrinter {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.out.println("Illegal number of arguments:" + args.length + ". Terminating.");
            System.exit(-1);
        }
        System.out.println("Upper Case Letters:");
        int index = 0;
        while (index < args[0].length()) {
            if (isUpperCase(args[0].charAt(index))) {
                System.out.print(args[0].charAt(index));
            }
        }
    }
}
```

Console

AnUpperCasePrinter [Java Application] C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Aug 30, 2005 4:57:50 PM)

Upper Case Letters: ← println()

STEP OVER

Debug - AnUpperCasePrinter.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Debug

AnUpperCasePrinter [Java Application]

- warmup.AnUpperCasePrinter at localhost:1171
 - Thread [main] (Suspended)
 - AnUpperCasePrinter.main(String[]) line: 12
 - C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Aug 30, 2005 4:57:50 PM)

Variables

- args= String[1] (id=11)
 - [0]= "John F. Kennedy"
- index= 0

[John F. Kennedy]

Outline

- warmup
 - AnUpperCasePrinter
 - main(String)
 - isUpperCas

```
System.out.println("Upper Case Letters:");
int index = 0;
while (index < args[0].length()) {
    if (isUpperCase(args[0].charAt(index)))
        System.out.print(args[0].charAt(index));

    index++;
}
System.out.println();

public static boolean isUpperCase(char c) {
    return (c >= 'A') && (c <= 'Z');
}
```

Console

AnUpperCasePrinter [Java Application] C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Aug 30, 2005 4:57:50 PM)

Upper Case Letters:

STEP INTO

Debug - AnUpperCasePrinter.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Debug

AnUpperCasePrinter [Java Application]

- warmup.AnUpperCasePrinter at localhost:1267
 - Thread [main] (Suspended)
 - AnUpperCasePrinter.isUpperCase(char) line: 19
 - AnUpperCasePrinter.main(String[]) line: 11
- C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Aug 30, 2005 8:24:05 PM)

Call stack

Variables

c = J

Called method parameters

[John F. Kennedy]

Outline

- warmup
 - AnUpperCasePrinter
 - main(String)
 - isUpperCase

```
if (isUpperCase(args[0].charAt(index)))  
    System.out.print(args[0].charAt(index));  
  
    index++;  
}  
System.out.println();  
  
public static boolean isUpperCase(char c) {  
    return (c >= 'A') && (c <= 'Z');  
}
```

Console

AnUpperCasePrinter [Java Application] C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Aug 30, 2005 8:24:05 PM)

Upper Case Letters:

STEP RETURN

Debug - AnUpperCasePrinter.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Debug

AnUpperCasePrinter [Java Application]

- warmup.AnUpperCasePrinter at localhost:1267
 - Thread [main] (Suspended)
 - AnUpperCasePrinter.main(String[]) line: 11
- C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Aug 30, 2005 8:24:05 PM)

Variables

- args= String[1] (id=11)
 - [0]= "John F. Kennedy"
- index= 0

[John F. Kennedy]

Outline

- warmup
 - AnUpperCasePrinter
 - main(String[])
 - isUpperCase(c)

```
public static void main(String[] args) {  
    if (args.length != 1) {  
        System.out.println("Illegal number of arguments:" + args.length + ". Termi  
        System.exit(-1);  
    }  
    System.out.println("Upper Case Letters:");  
    int index = 0;  
    while (index < args[0].length()) {  
        if (isUpperCase(args[0].charAt(index)))  
            System.out.print(args[0].charAt(index));  
  
        index++;  
    }  
}
```

Console

AnUpperCasePrinter [Java Application] C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Aug 30, 2005 8:24:05 PM)

Upper Case Letters:

RESUME

Debug - AnUpperCasePrinter.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Debug Console

<terminated> AnUpperCasePrinter [Java Application]
<terminated> warmup.AnUpperCasePrinter at localhost:1267
<terminated, exit value: 0> C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Aug 30, 20

Variables Breakpoints

Outline

```
public static void main(String[] args) {  
    if (args.length != 1) {  
        System.out.println("Illegal number of arguments:" + args.length + ". Termi  
        System.exit(-1);  
    }  
    System.out.println("Upper Case Letters:");  
    int index = 0;  
    while (index < args[0].length()) {  
        if (isUpperCase(args[0].charAt(index)))  
            System.out.print(args[0].charAt(index));  
  
        index++;  
    }  
}
```

Console Tasks

<terminated> AnUpperCasePrinter [Java Application] C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Aug 30, 2005 8:24:05 PM)
Upper Case Letters:
JFK

RETURNING TO JAVA PERSPECTIVE

Java - AnUpperCasePrinter.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Hierarchy

- examples
 - warmup
 - AHelloWorldGreeter.java
 - AnUpperCasePrinter.java
 - ASafeArgPrinter.java
 - JRE System Library [jre1.5.0_04]

```
package warmup;
public class AnUpperCasePrinter {
    public static void main(String[] args){
        if (args.length != 1) {
            System.out.println("Illegal number of arguments:");
            System.exit(-1);
        }
        System.out.println("Upper Case Letters:");
        int index = 0;
        while (index < args[0].length()) {
            if (isUpperCase(args[0].charAt(index)))
                System.out.print(args[0].charAt(index));

            index++;
        }
        System.out.println();
    }
    public static boolean isUpperCase(char c) {
        return (c >= 'A') && (c <= 'Z');
    }
}
```

Outline

- warmup
 - AnUpperCasePrinter
 - main(String[])
 - isUpperCase(char)

Problems Javadoc Declaration Console

<terminated> AnUpperCasePrinter [Java Application] C:\Program Files\Java\jre1.5.0_04\bin\javaw.exe (Aug 30, 2005 8:24:05 PM)

Upper Case Letters:
JFK

warmup - examples Writable Smart Insert 11 : 1

DEBUG LAST MAIN LAUNCHED: F11

No need to select a
main class

CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- **Adding a library**
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

LIBRARY LOCATION ON THE WEB

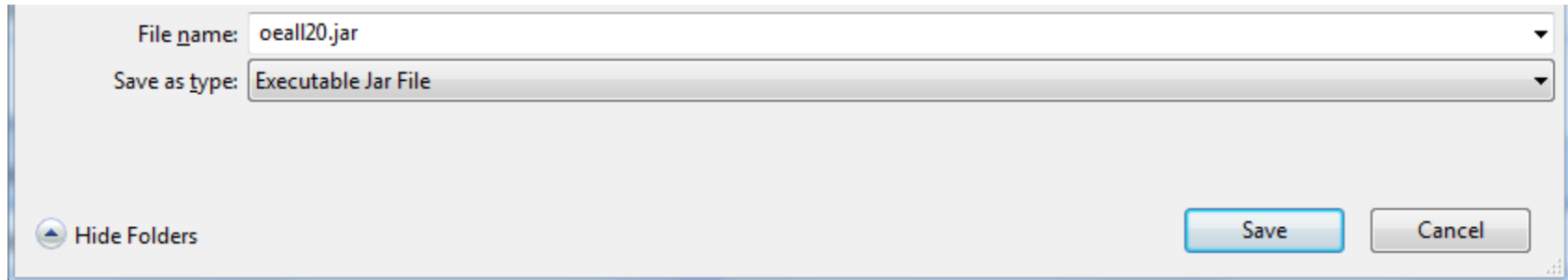
Downloads

Downloads

ObjectEditor Version 3 (used in comp110)	oeall3
ObjectEditor Version 19 (used last year)	oeall19
ObjectEditor Version 21	oeall21
ObjectEditor Version 22 (latest, use this unless it fails on you)	oeall22
Checkstyle Jar	UNCChecks_6.5.0.jar

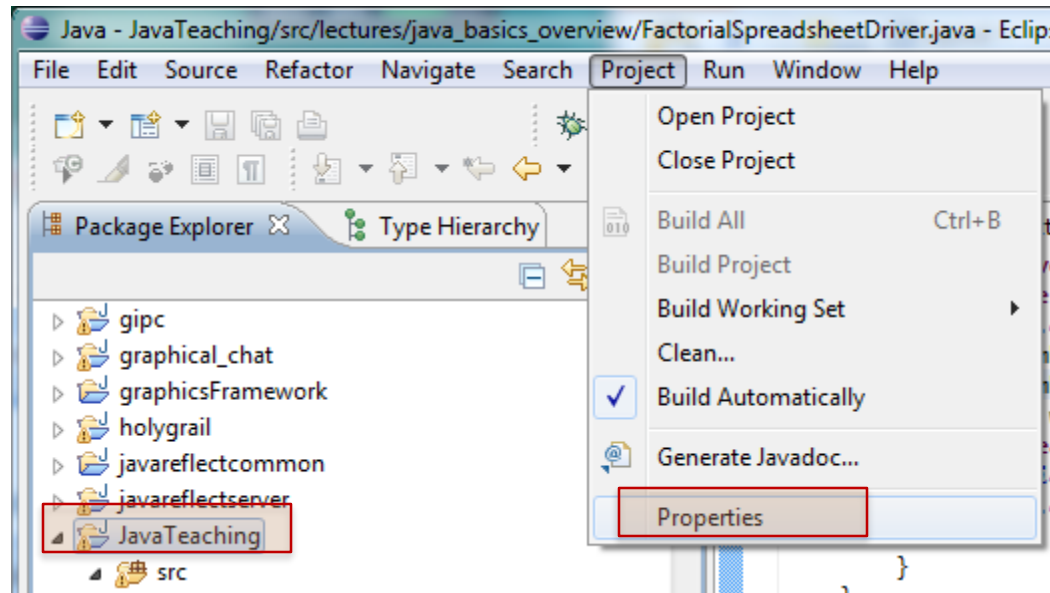
Click on the latest version (on my web page, not this page) to download the jar (same as zip) and do not unzip/unjar it

DOWNLOAD LIBRARY TO LOCAL COMPUTER



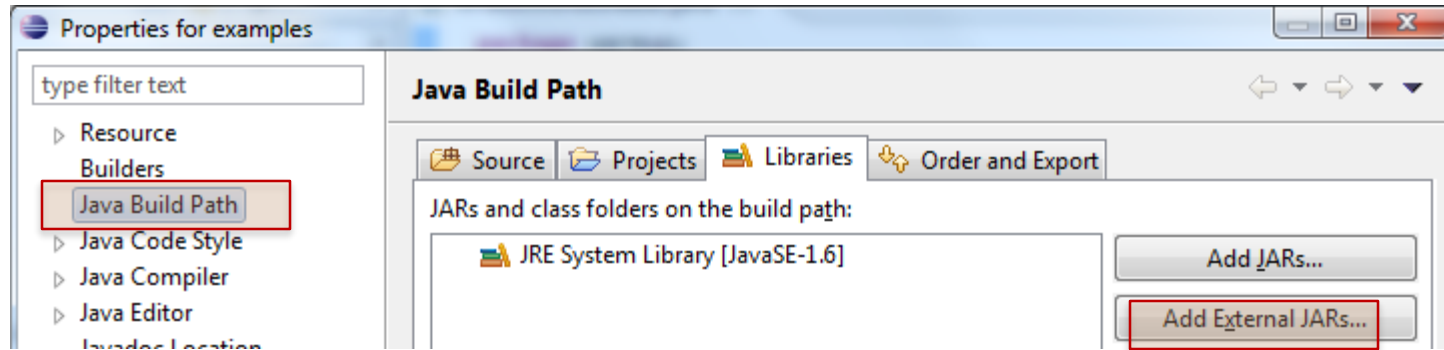
Do not unzip/unjar it

SELECT PROJECT, AND PROJECT→PROPERTIES

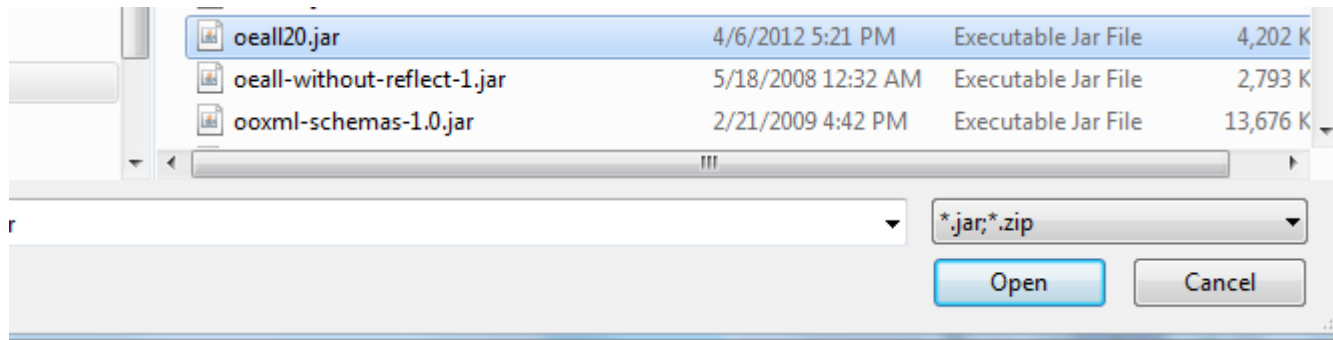


Must do this for each project!

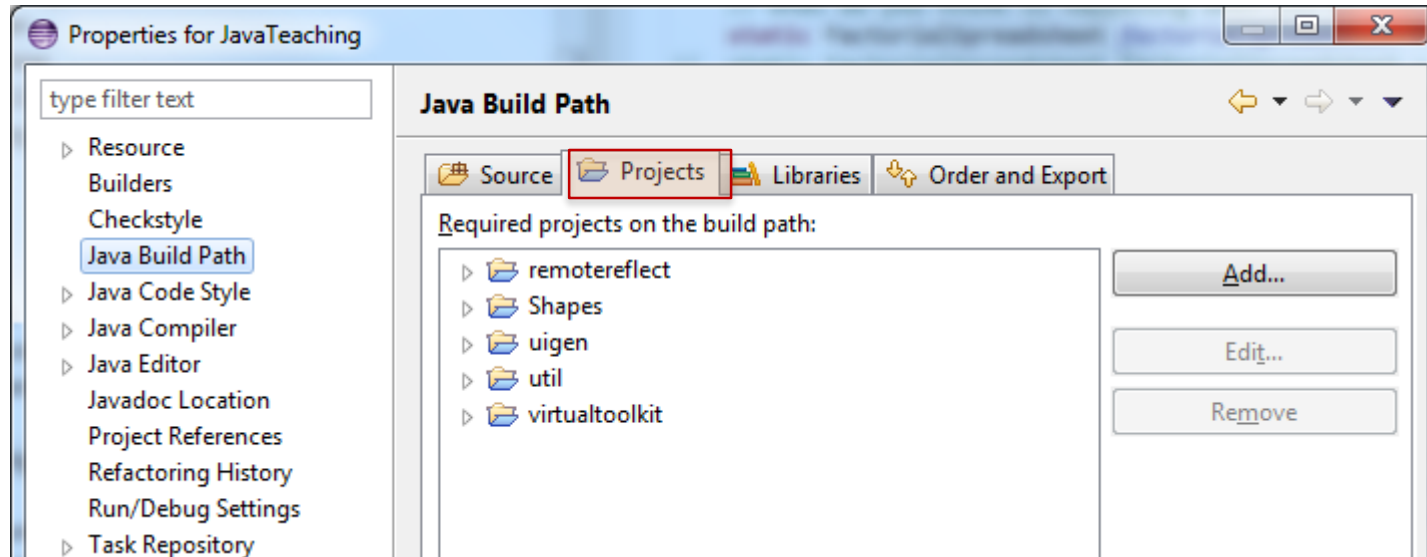
SELECT BUILD PATH, ADD EXTERNAL JAR



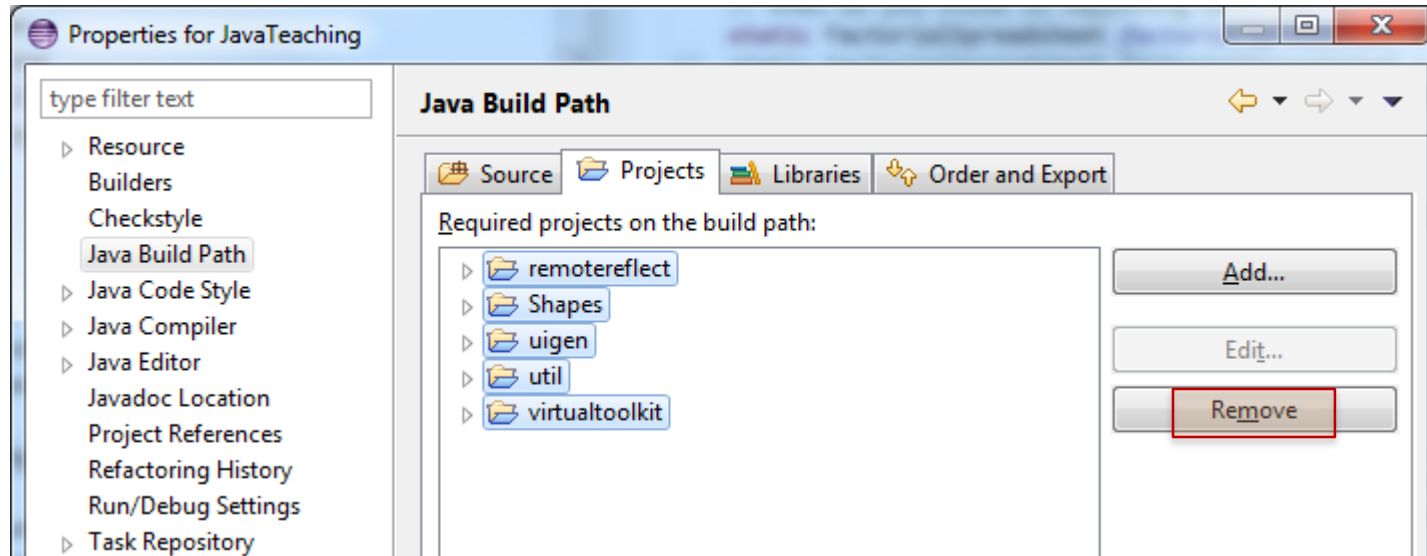
SELECT SAVED JAR



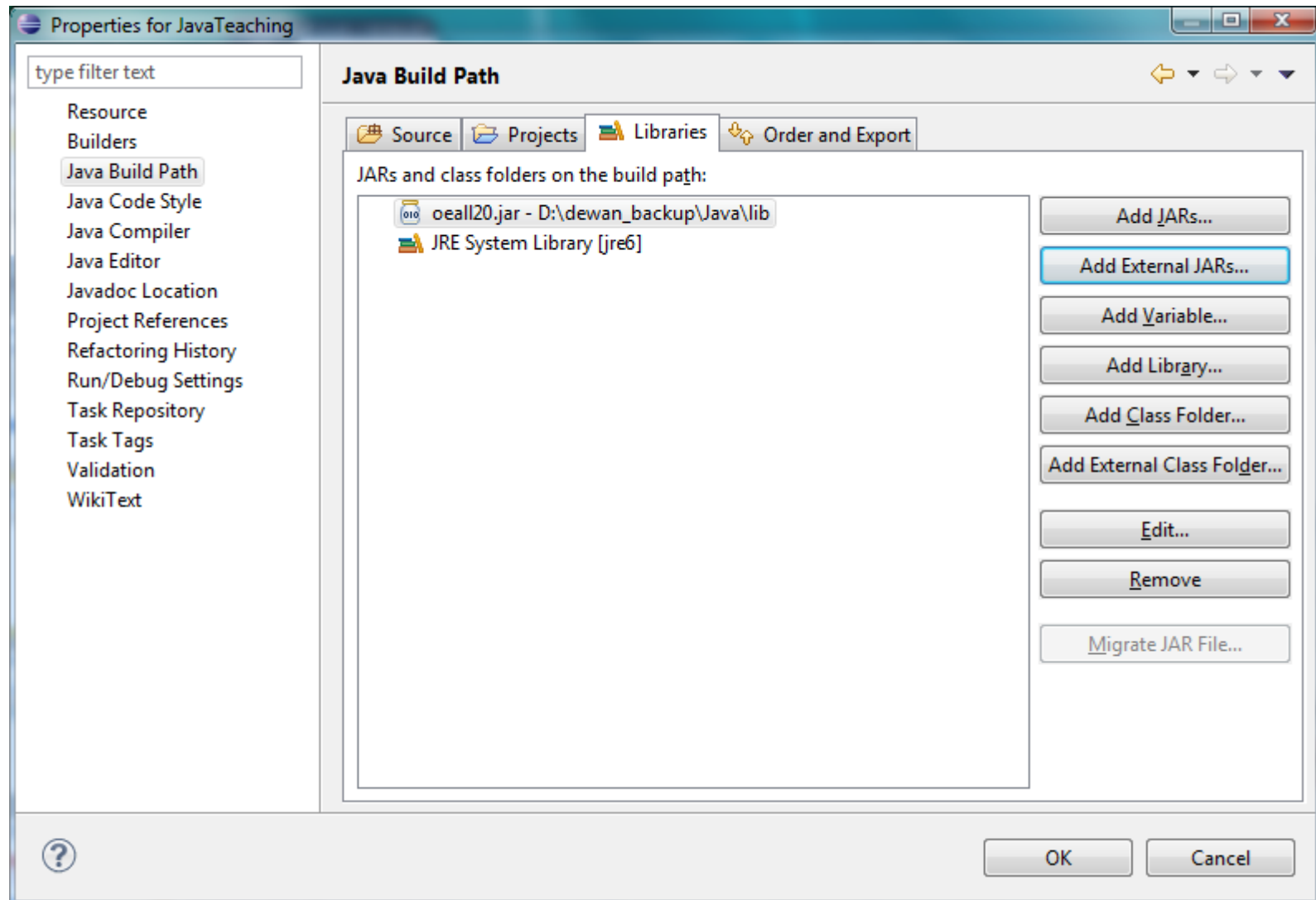
SELECT BUILD PATH, PROJECTS



REMOVE SPURIOUS PROJECTS (IF ANY)



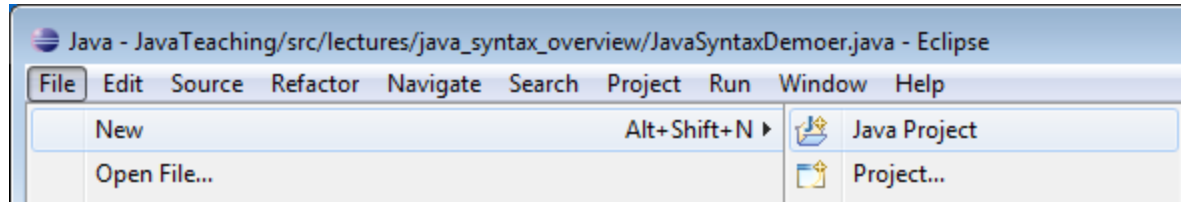
PRESS OK TO COMMIT



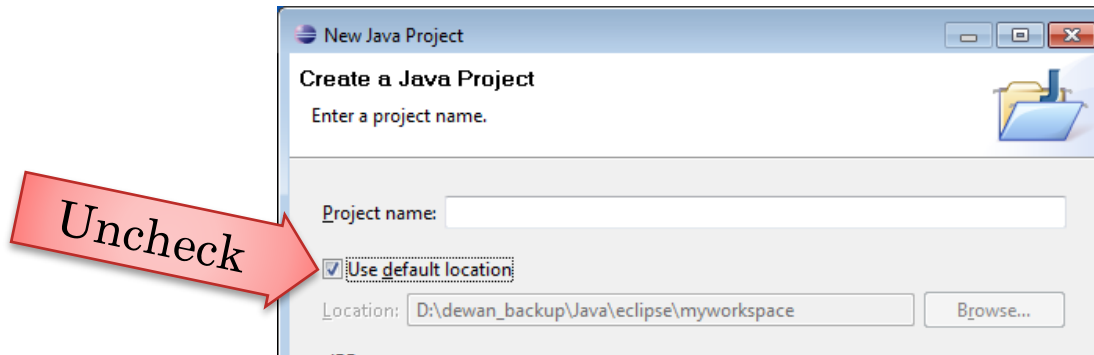
CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- **Add an existing project**
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

CREATE A PROJECT OUT OF EXISTING CODE: FILE→NEW→JAVA PROJECT



UNCHECK DEFAULT LOCATION BOX



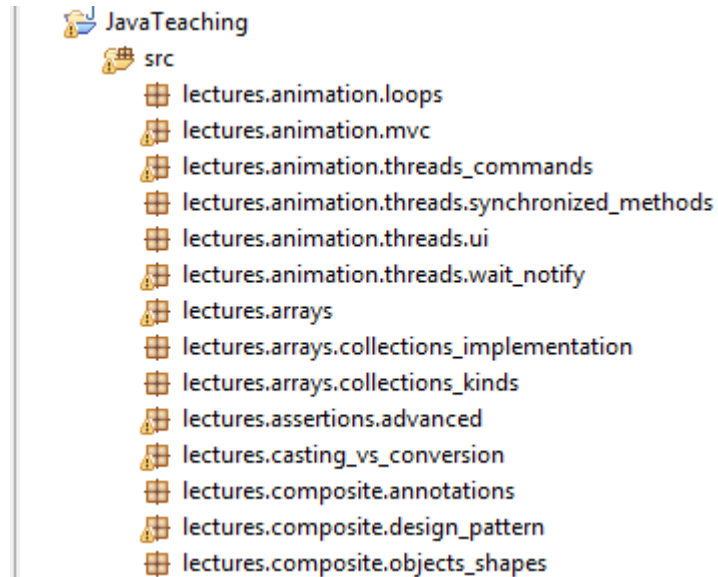
ENTER LOCATION AND NAME

Project name:

☐ Use default location

Location:

NEW PROJECT CREATED



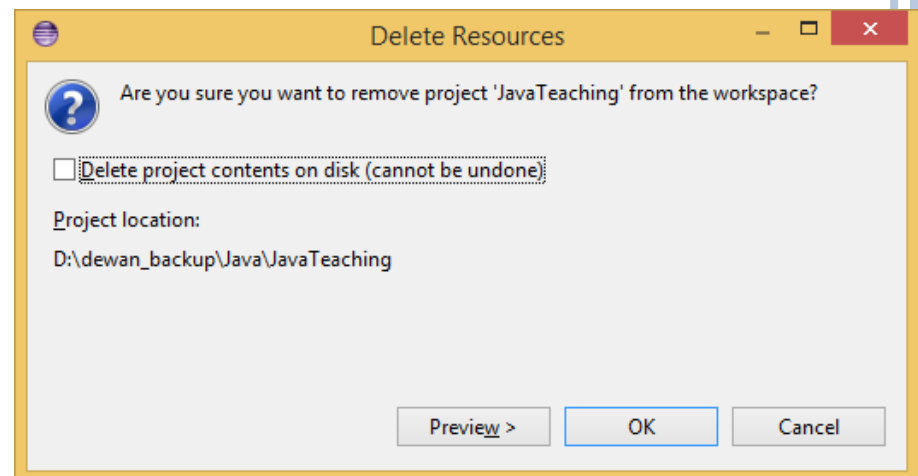
But it will have errors because of missing
ObjectEditor library

CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- **Copy a project**
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

COPYING A PROJECT

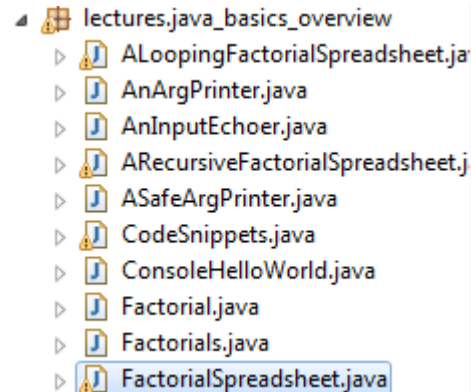
Select project, CTRL CTRL-V



CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- **Copy a package, class file to another project**
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

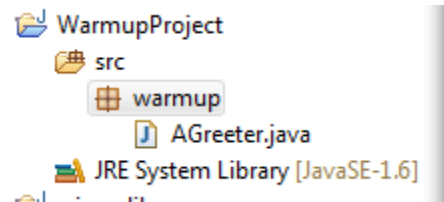
COPY TYPES (CLASSES, INTERFACES, ENUMS) FROM PACKAGE TO ANOTHER: COPY



```
package lectures.java_basics_overview;  
public interface FactorialSpreadsheet {  
    public int getNumber();  
    public void setNumber(int newVal) ;  
    public long getFactorial();  
}
```

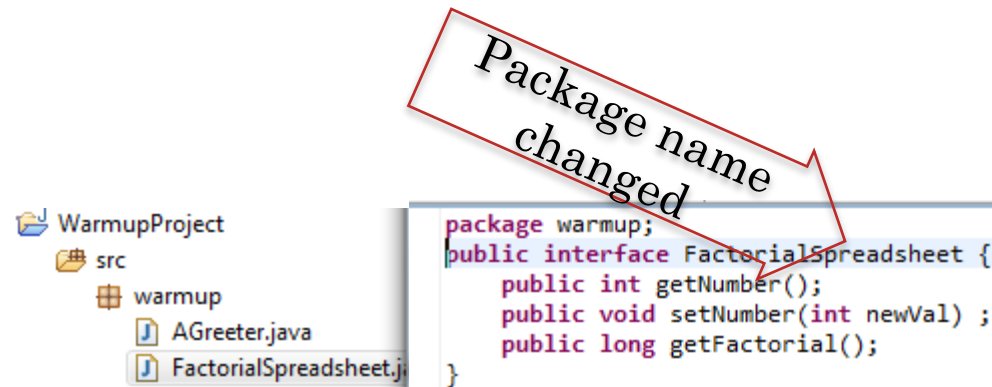
Select type(s) in source project
and package and press CTRL-C
(or File→Copy right menu
item)

COPY TYPES FROM ONE PACKAGE TO ANOTHER: PASTE



Select package in destination
project and package and press
CTRL-V (or File→Paste right
menu item)

COPYING CLASSES FROM ONE PACKAGE TO ANOTHER: NEW CLASS IN NEW PACKAGE AND PROJECT

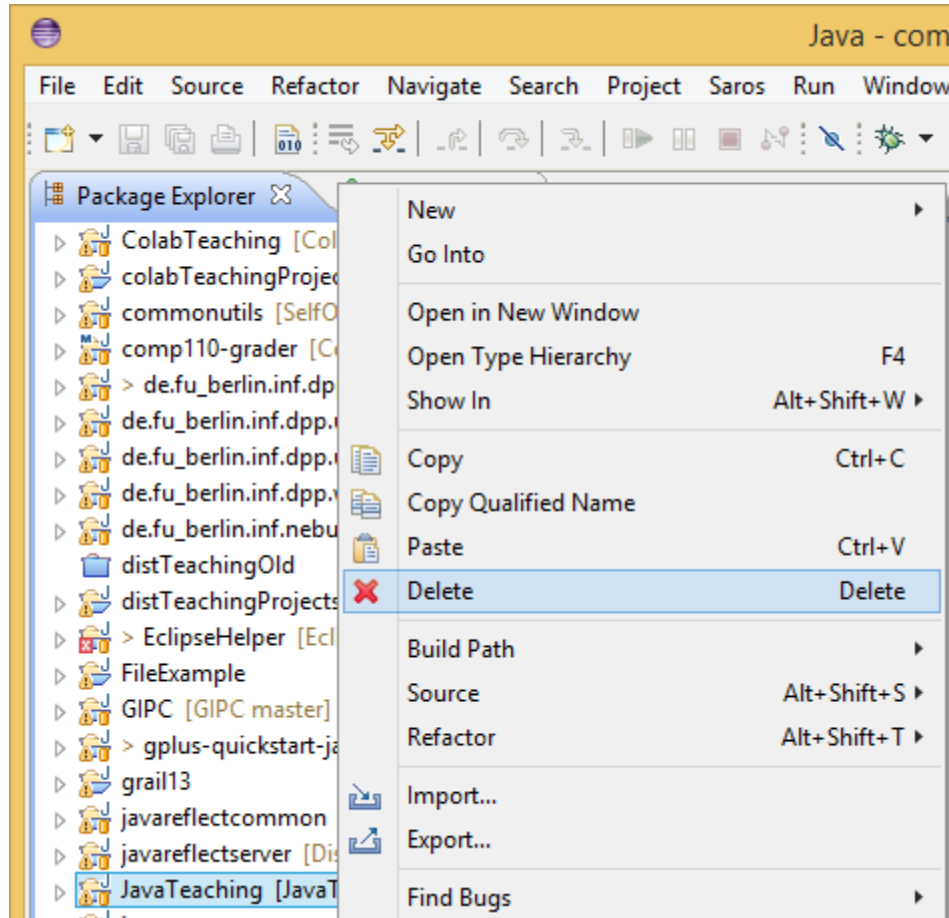


Repackaged type added (and its imports are changed to other copied types if multiple types are copied and pasted together)

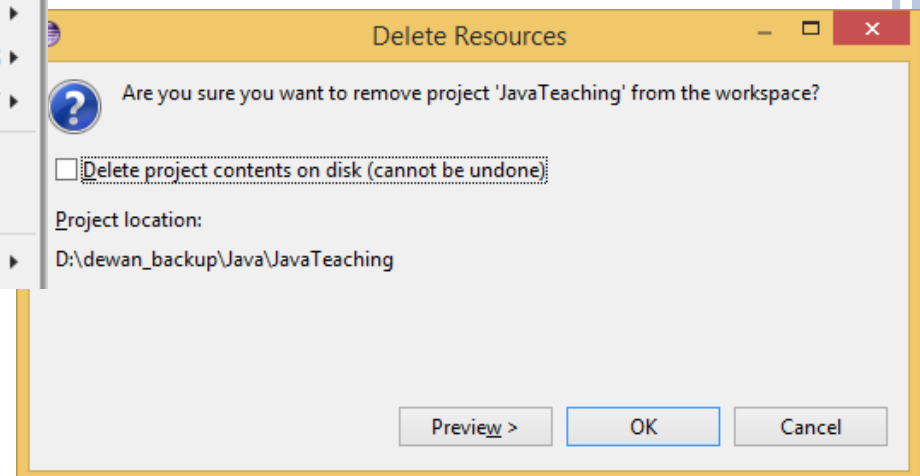
CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- **Delete a project**
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

DELETING A PROJECT



Right click project and delete.
Usually not a good idea to delete project on disk, can always use the OS to do so, which will put it in the recycling bin



CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- **Show variable, class, interface (identifier)** definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

JUMPING TO IDENTIFIER DEFINITION AND NAVIGATION STACK

Cursor
position

```
FactorialSpreadsheet factorialSpreadsheet = new ARecursiveFactorialSpreadsheet();
```

ALT + Forward Arrow
goes forwards in the
navigation stack a la
browser forward
button

ALT + Back Arrow takes
you back where you
came from, a la
Browser back button,
can execute multiple
times to back in in the
stack of visited points

F3 when cursor is a start
of an identifier

Makes Eclipse take you to
the definition of the
identifier in an edit window

```
    long factorial;  
    public int getNumber() {  
        return number;  
    }  
    public void setNumber(int newVal) {  
        number = newVal ;  
    }  
    public long getFactorial() {  
        return Factorials.recurringFactorial(number);  
    }  
    public static int factorial(int n) {  
        if (n <= 1)  
            return 1;  
        return n*factorial(n-1);  
    }  
}
```

CONTENTS

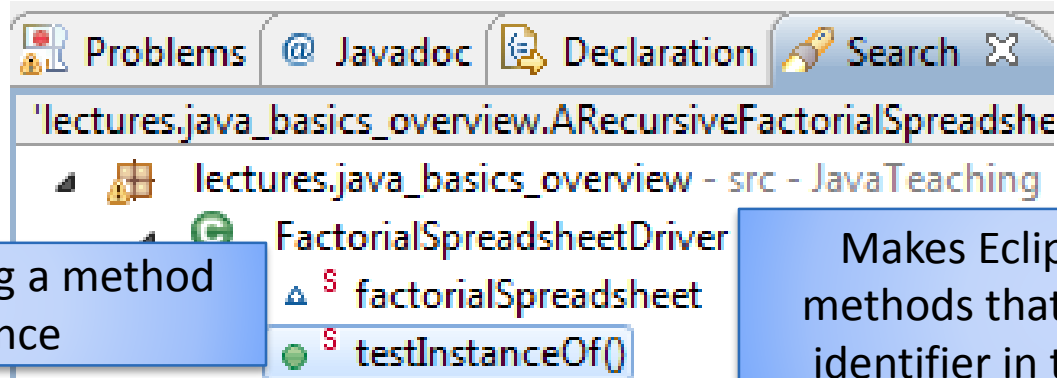
- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- **Find identifier (variable, class, interface) use**
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

FIND THE USES

```
public class ARecursiveFactorialSpreadsheet implements FactorialSpreadsheet {  
    int n;  
    long factorial;
```

CTRL-SHIFT-G when cursor is a
start of an identifier

Cursor position



Double clicking a method
reference

Makes Eclipse show the
methods that reference the
identifier in the search tab

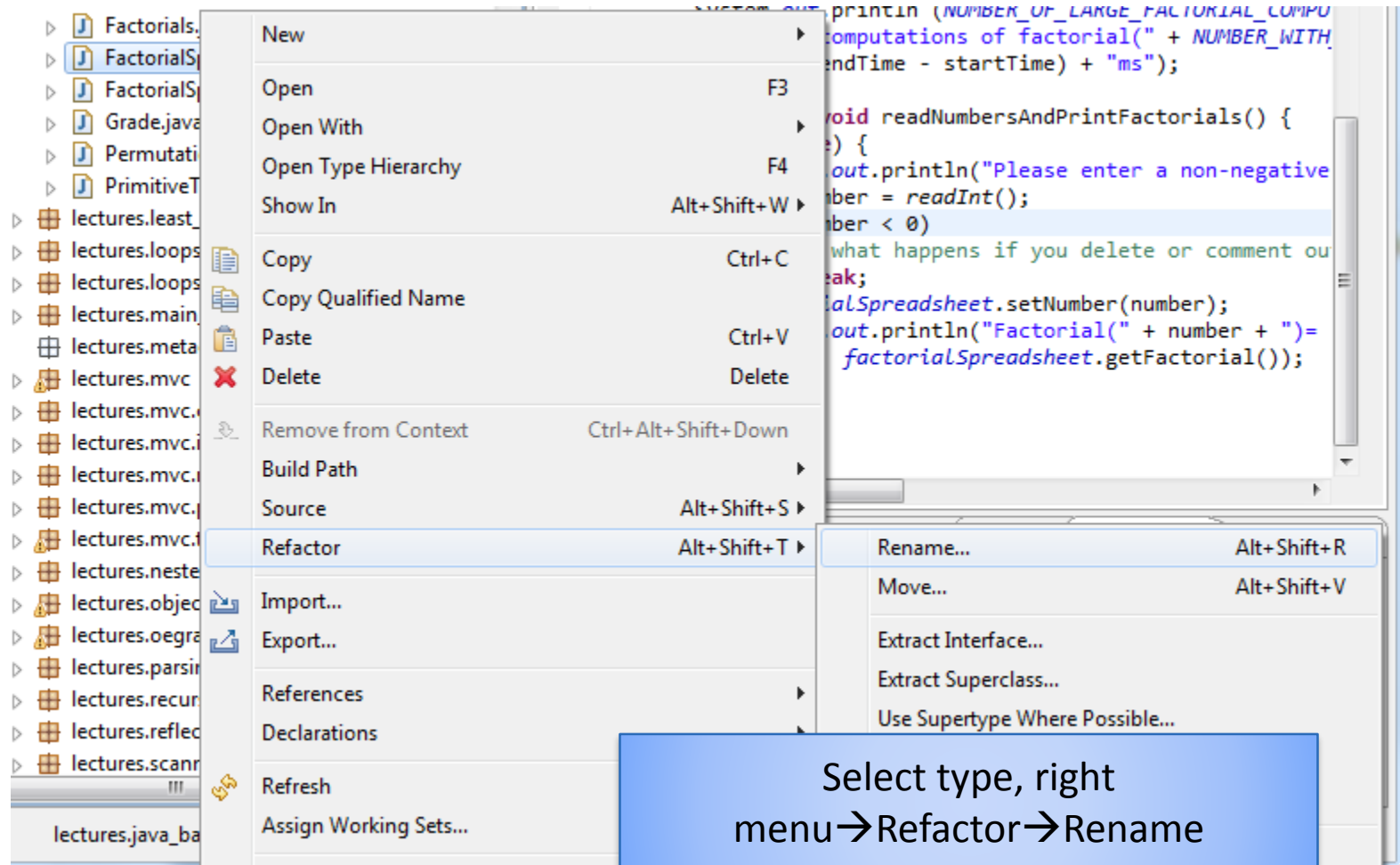
```
static FactorialSpreadsheet factorialSpreadsheet = new ARecursiveFactorialSpreadsheet();
```

Makes Eclipse show the method
and highlight all references of
the identifier

CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- **Refactor**
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

MENU → REFACTOR → RENAME IDENTIFIER: ALT + SHIFT + R



Select type, right
menu → Refactor → Rename

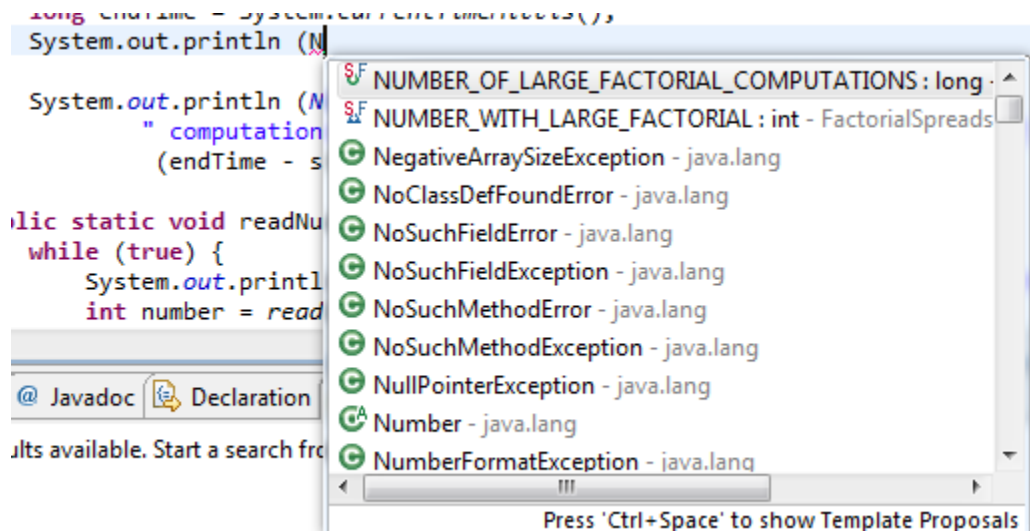
All references will be updated!

CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- **Complete identifier name**
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

COMPLETE IDENTIFIER: CTRL + SPACE AFTER PREFIX

```
public static final long NUMBER_OF_LARGE_FACTORIAL_COMPUTATIONS  
= 1000000;
```

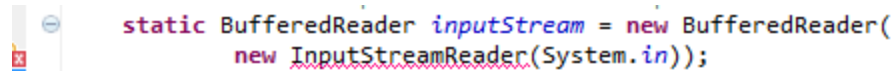


Give file name

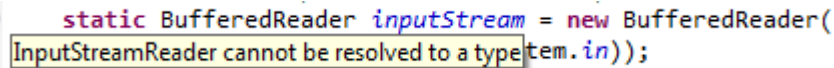
CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- **Understand, fix errors**
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface

UNDERSTAND AND CORRECT ERRORS



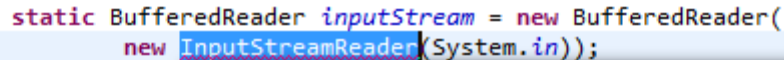
```
static BufferedReader inputStream = new BufferedReader(  
    new InputStreamReader(System.in));
```



```
static BufferedReader inputStream = new BufferedReader(  
    InputStreamReader cannot be resolved to a type inputStream.in);
```

Hover to
understand

Click for
correction
proposal



```
static BufferedReader inputStream = new BufferedReader(  
    new InputStreamReader(System.in));
```

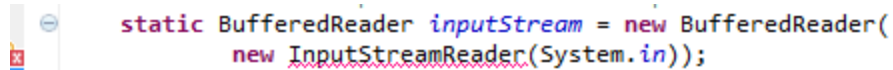
- Import 'InputStreamReader' (java.io)
- Create class 'InputStreamReader'
- Change to 'Reader' (java.io)
- Change to 'InputStream' (java.io)
- Change to 'InputStream' (org.omg.CORBA.portable)
- Change to 'InputStream' (org.omg.CORBA_2_3.portable)
- Change to 'InputStreamAdapter' (com.sun.imageio.plugins.jpeg)
- Change to 'InputStreamHook' (com.sun.corba.se.impl.io)
- Change to 'InputStreamImageSource' (sun.awt.image)
- Rename in file (Ctrl+2, R)
- Fix project setup...

Problems @ Javadoc

CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- **Automatic imports**
- JUnit
- Change Method Signature
- Extract interface

CORRECT ALL IMPORT ERRORS IN FILE: CTRL + SHIFT + O



```
static BufferedReader inputStream = new BufferedReader(  
    new InputStreamReader(System.in));
```

In Eclipse press CTRL-SHIFT-O to automatically import all used types that need to be imported but have not been imported

If class is in more than one package, Eclipse gives a choice

All unused imports are also removed

Can select multiple types or a project to refresh all imports in the selected types/project

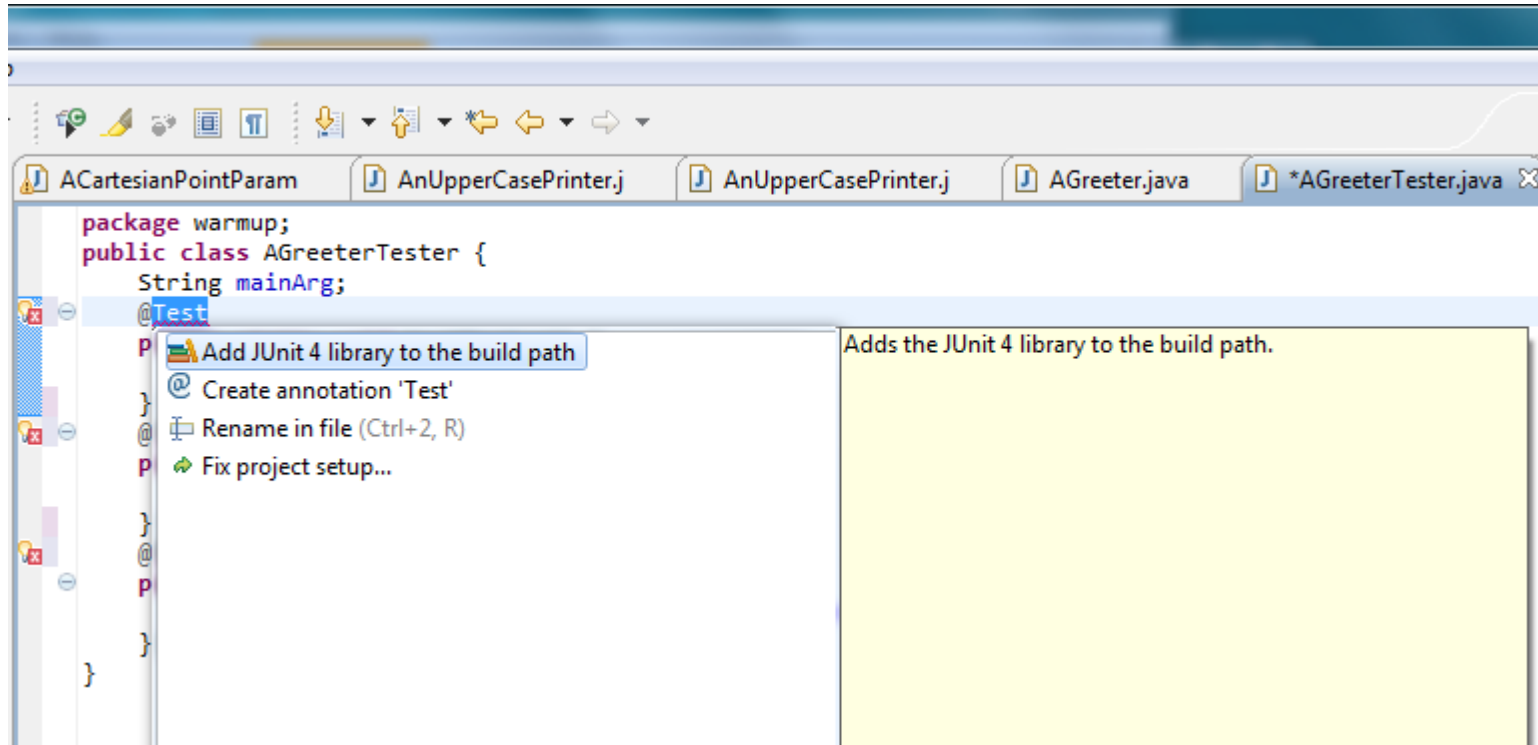
CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- **JUnit Testing**
- Change Method Signature
- Extract interface

JUNIT: NEW CLASS WITHOUT MAIN

```
package warmup;
public class AGreeterTester {
    String mainArg;
    @Test
    public void testOneArg() {
        AGreeter.main(new String[]{"Ca Va"});
    }
    @Test
    public void tesNoArg() {
        AGreeter.main(new String[]{});
    }
    @Test
    public void tesMultipleArgs() {
        AGreeter.main(new String[]{"Ca Va", "Buon Giorno"});
    }
}
```

JUNIT: INSTALL JUNIT



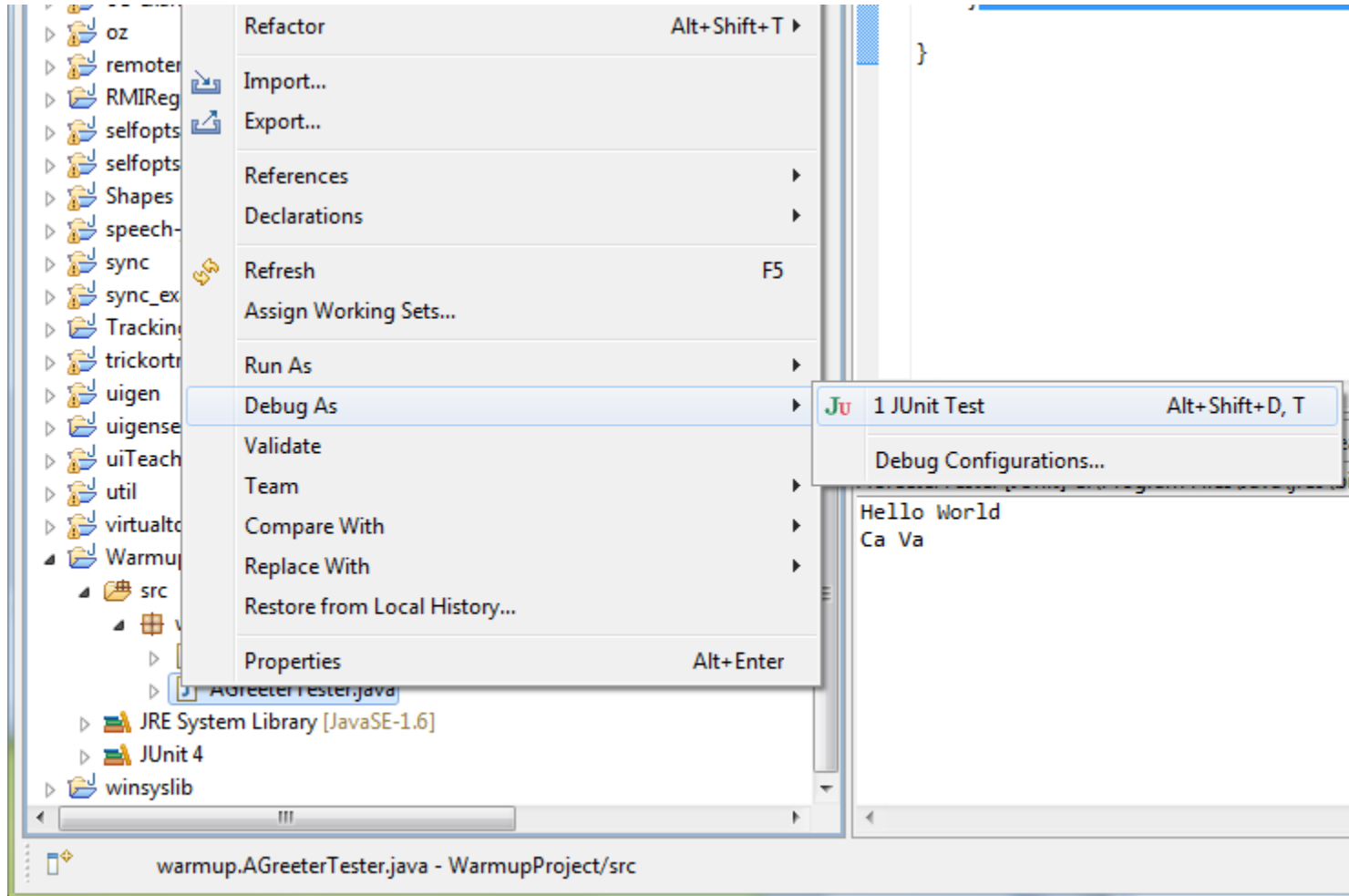
JUNIT ADDED AND IMPORTED

```
package warmup;

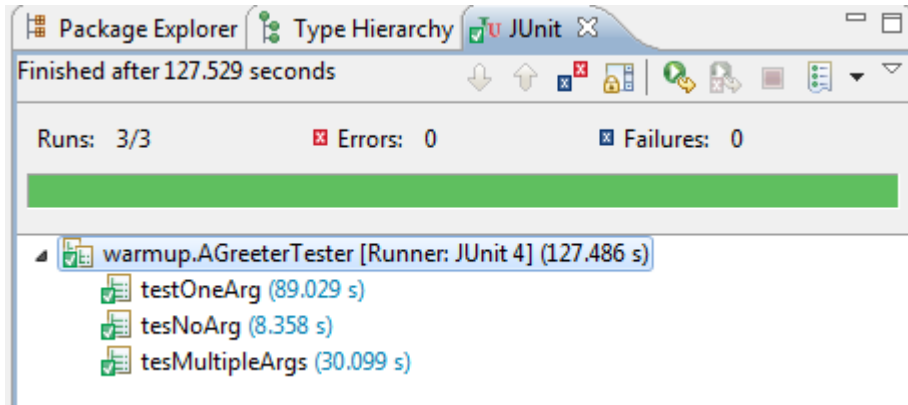
import org.junit.Test;

public class AGreeterTester {
    String mainArg;
    @Test
    public void testOneArg() {
        AGreeter.main(new String[]{"Ca Va"});
    }
    @Test
    public void tesNoArg() {
        AGreeter.main(new String[]{});
    }
    @Test
    public void tesMultipleArgs() {
        AGreeter.main(new String[]{"Ca Va", "Buon Giorno"});
    }
}
```

RUN JUNIT

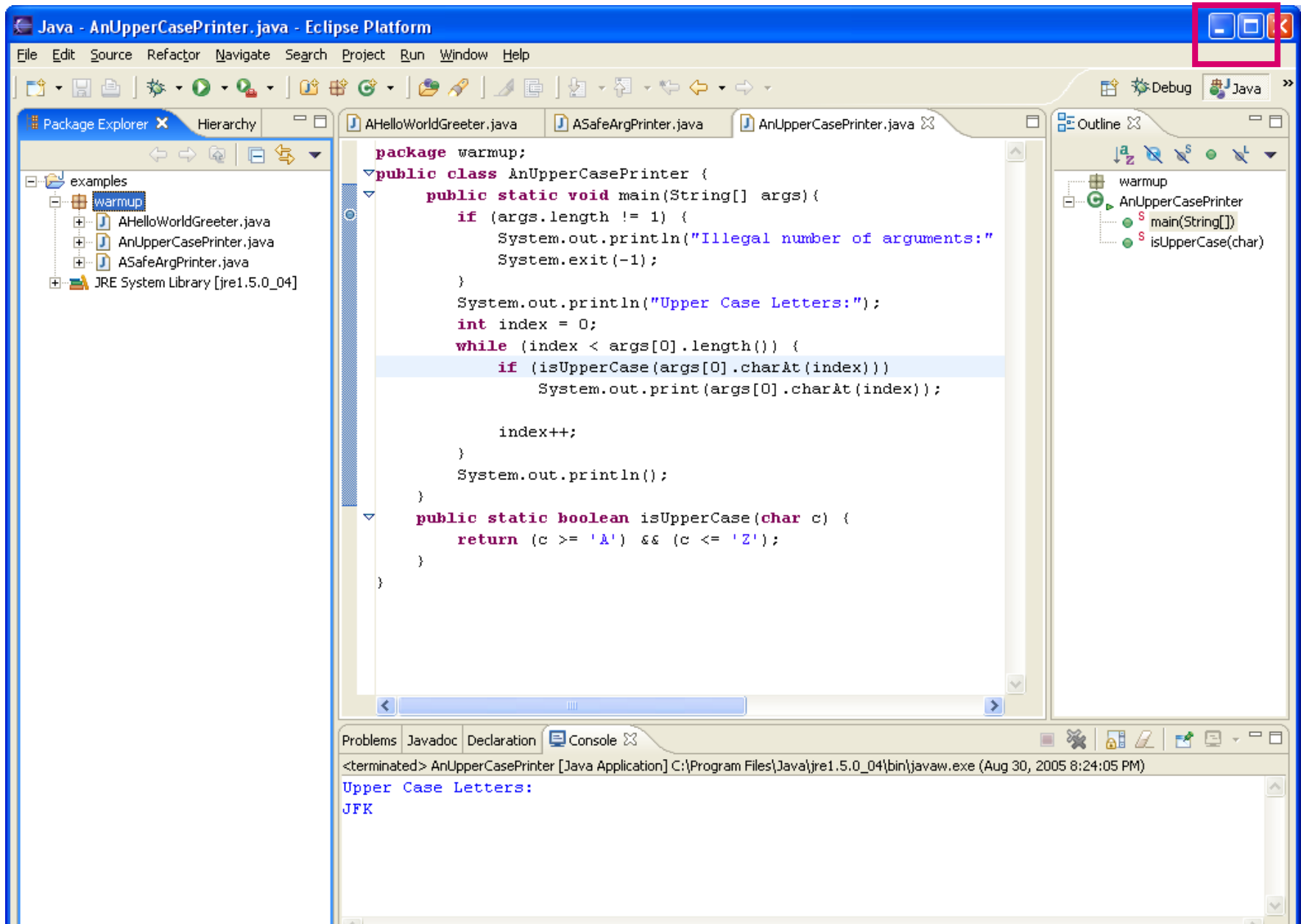


JUNIT RESULT



```
<terminated> AGreeterTester [JUnit]
Ca Va
Ni hao
Ni hao
Hello World
Namaste
Namaste
Hello World
Ca Va
Guten Tag
Guten Tag
```

RETURNING TO JAVA PERSPECTIVE



CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- **Change Method Signature**
- Extract interface

TWO OTHER USEFUL BUT COMPLICATED REFACTOR OPTIONS

Change method signature

Will try to update references to match the signature, which can be erroneous

Extract interface of class C

Will create interface of selected public methods and will make the interface the type of all variables declared to be of class type

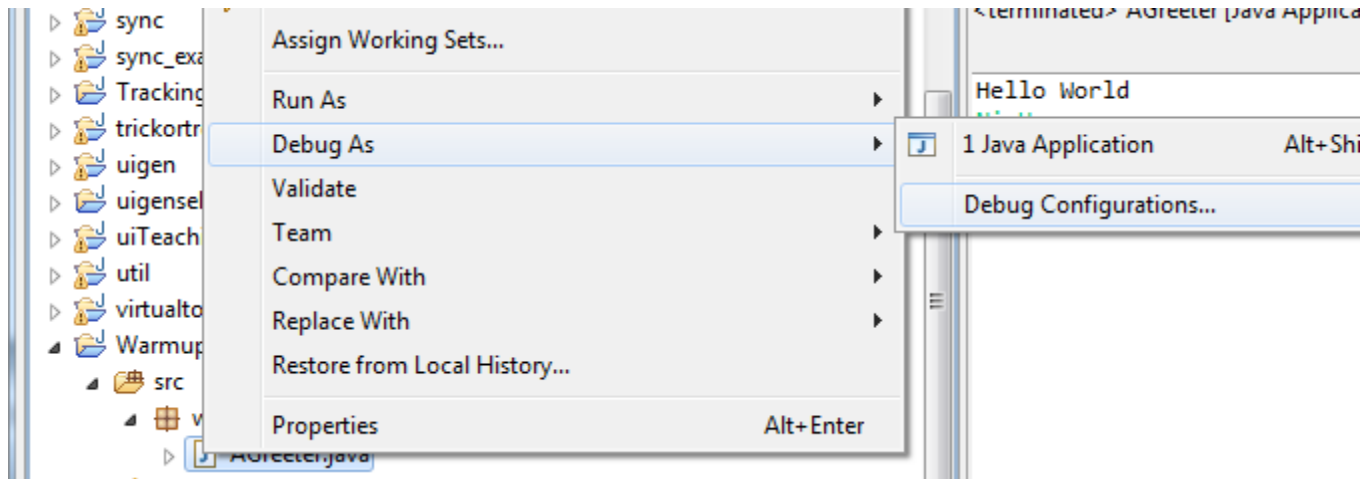
Adds potentially unwanted comments to method headers and sometimes updates to references have not worked in the past for me

Maybe better to get all references and update them individually for both purposes

CONTENTS

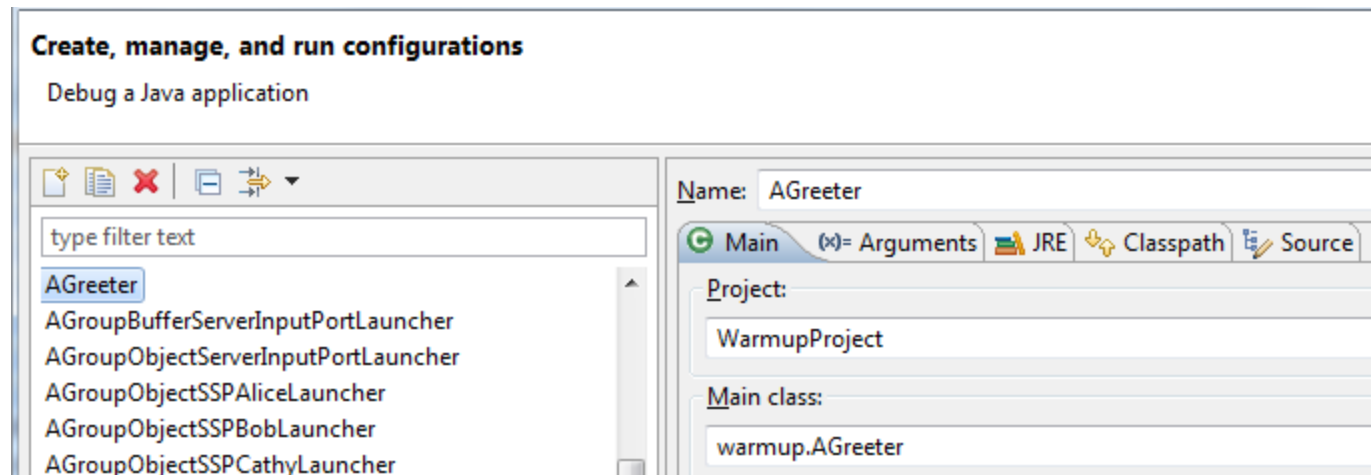
- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface
- **Supply main argument**

SUPPLY USER ARGUMENT:



Right Menu → Debug As → Debug Configuration

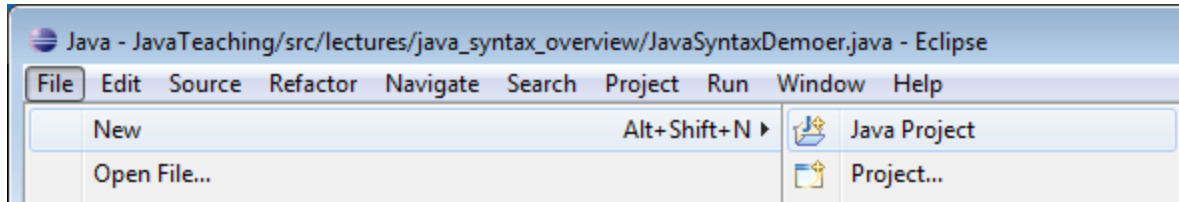
SELECT MAIN CLASS IN CONFIGURATION



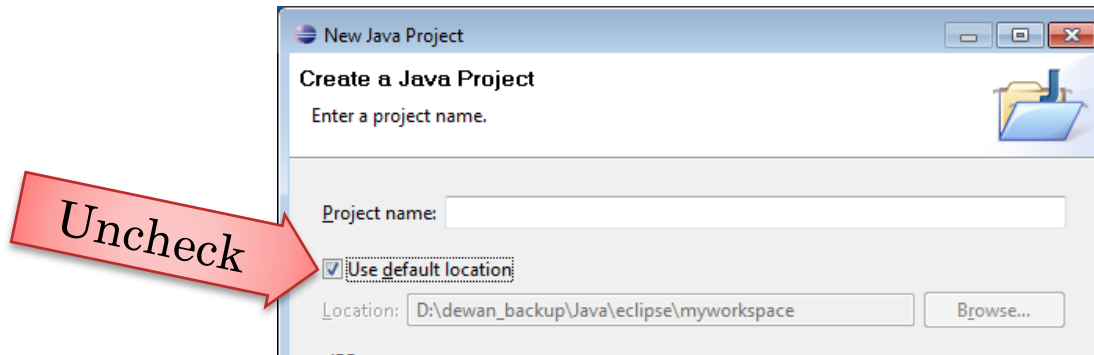
CONTENTS

- Install JDK
- Install Eclipse
- Connect JDK to Eclipse
- Create project
- Edit project run project
- Restore window
- Search project for strings
- Debugging
- Adding a library
- Add an existing project
- Copy a project
- Copy a package, class file to another project
- Delete a project
- Show variable, class, interface (identifier) definition and return back
- Find identifier (variable, class, interface) use
- Refactor
- Complete identifier name
- Understand, fix errors
- Automatic imports
- JUnit
- Change Method Signature
- Extract interface
- Supply main argument
- **Create project from existing code**

CREATE A PROJECT OUT OF EXISTING CODE: FILE→NEW→JAVA PROJECT



UNCHECK DEFAULT LOCATION BOX



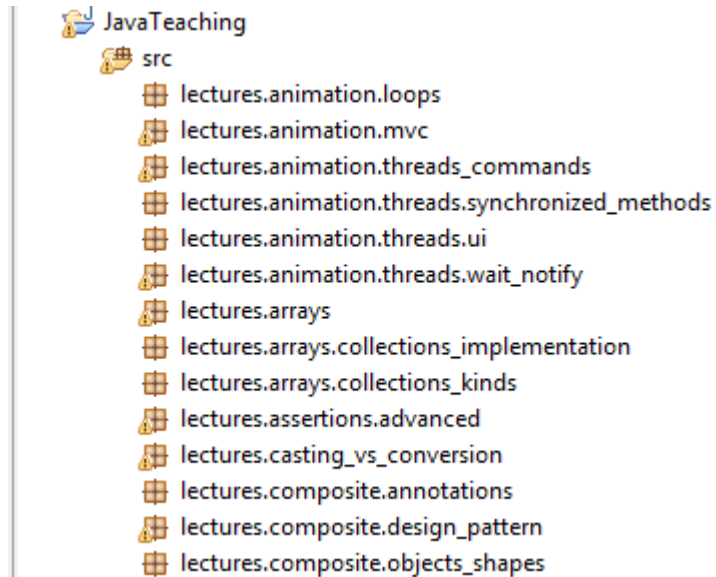
ENTER LOCATION AND NAME

Project name:

☐ Use default location

Location:

NEW PROJECT CREATED

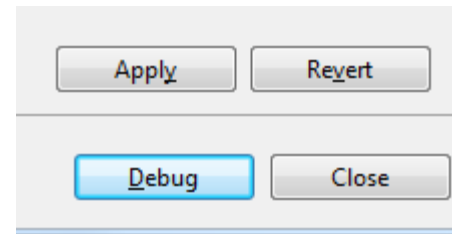
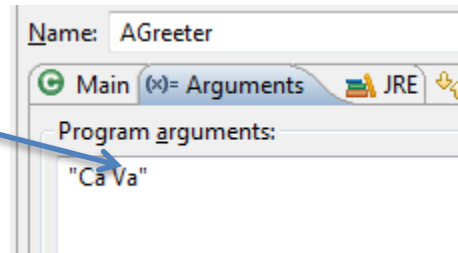


But it will have errors because of missing
ObjectEditor library

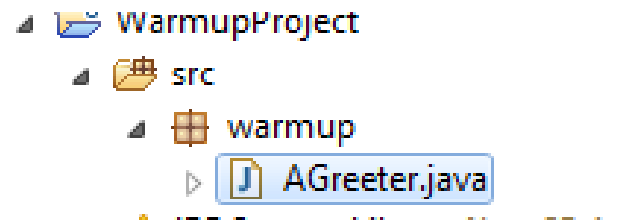
SPECIFY AND APPLY ARGUMENT IN ARGUMENT TAB AND PRESS DEBUG

Argument must be in quotes. Args[0] = "Ca Va"

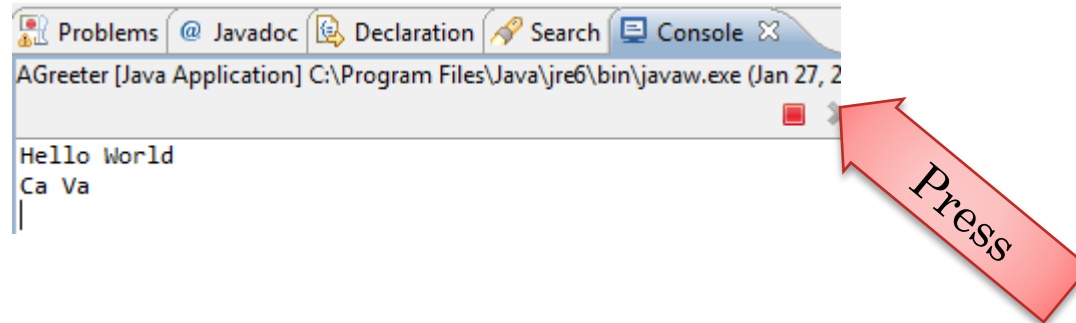
Without quotes Java will make each word a separate argument
(args[0] = "Ca", args[1] = "Va")



EXECUTE WITH SUPPLIED ARG



ARG PRINTED, PROGRAM, STILL RUNNING, WAITING FOR INPUT



Button available in
both Debug and Java
Perspective, only of
them is active

TERMINATED PROGRAM

