First Year Eclipse Tutorials

Copying a Class

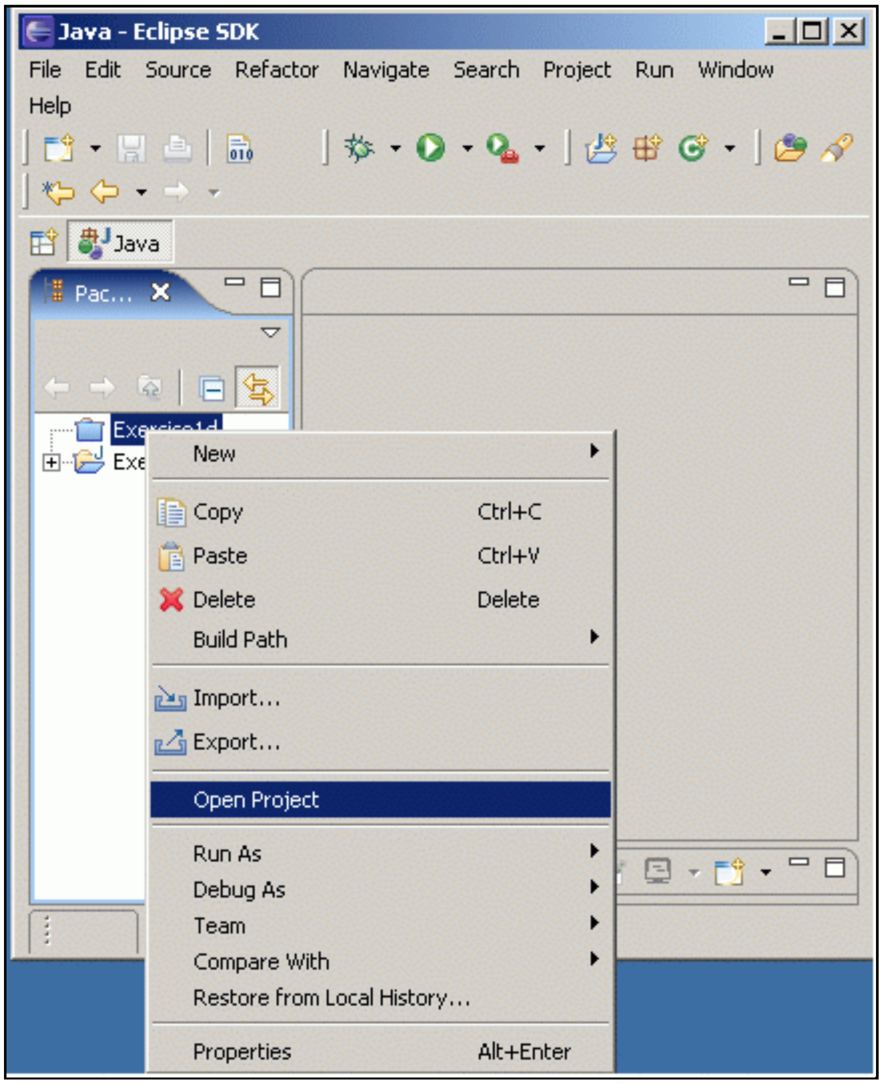
This tutorial will show you how to copy a class from another project to the current project, maintaining the same class name.

It applies to an older version of 114, so although specific file names and projects no longer apply, the concepts still hold.

You will be copying *Exercise1d.java* from project *Exercise1d* , to project *Exercise2b* . The name will remain *Exercise1d.java* .

1. Make a new project called *Exercise2b* , referring to the Application Tutorial for more information if needed.
   You will be copying files to Exercise2b.

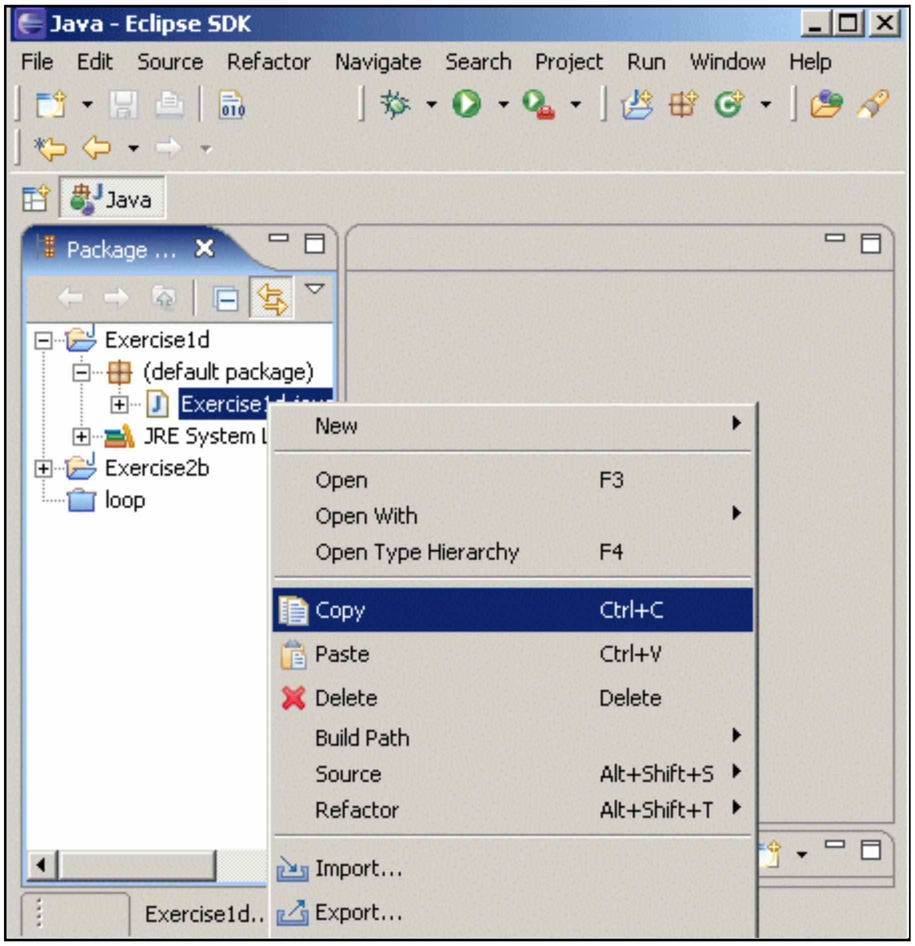
2. If it's not already open, open the *Exercise1d* project, by right-clicking on the *Exercise1d* project, and choosing *Open Project* from the context menu that appears.



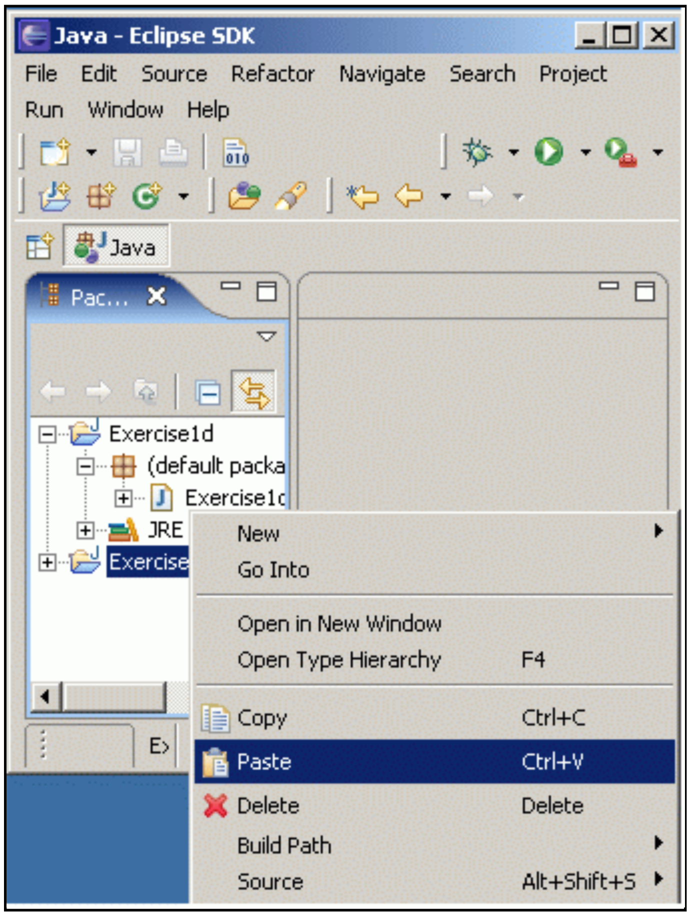
If prompted with *Should referenced projects also be opened where applicable?* , click *Yes*
You will be copying files from Exercise1d.

3. Right-click on *Exercise1d.java* , and choose copy from the context menu. If Exercise1d.java is not visible in the *Package Explorer* , you will first need to click the arrow (plus signs in the screenshot below will be arrows in the lab's version of Eclipse.) beside the *Exercise1d* project, then the arrow beside *src* (not shown in screenshot), and then the arrow beside *(default package)* .

   Note the plus signs in the screenshot below will be arrows in the lab's version of Eclipse.



4. Right-click on the src directory of the *Exercise2b* project (the screenshot does not show this), and choose *Paste* . Click on arrows appropriately if src is not visible.



5. Your code is now copied. To keep things simple, it is likely best to close the *Exercise1d* project by right-clicking on it, and choosing *Close Project* from the context menu.

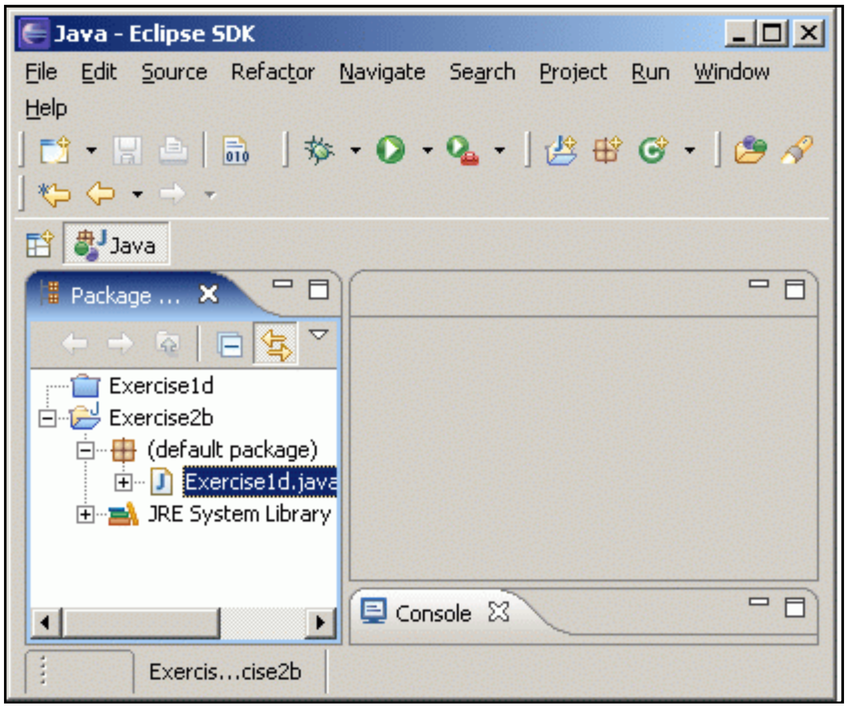
Renaming a Class

This tutorial continues from the previous Copying a Class tutorial.

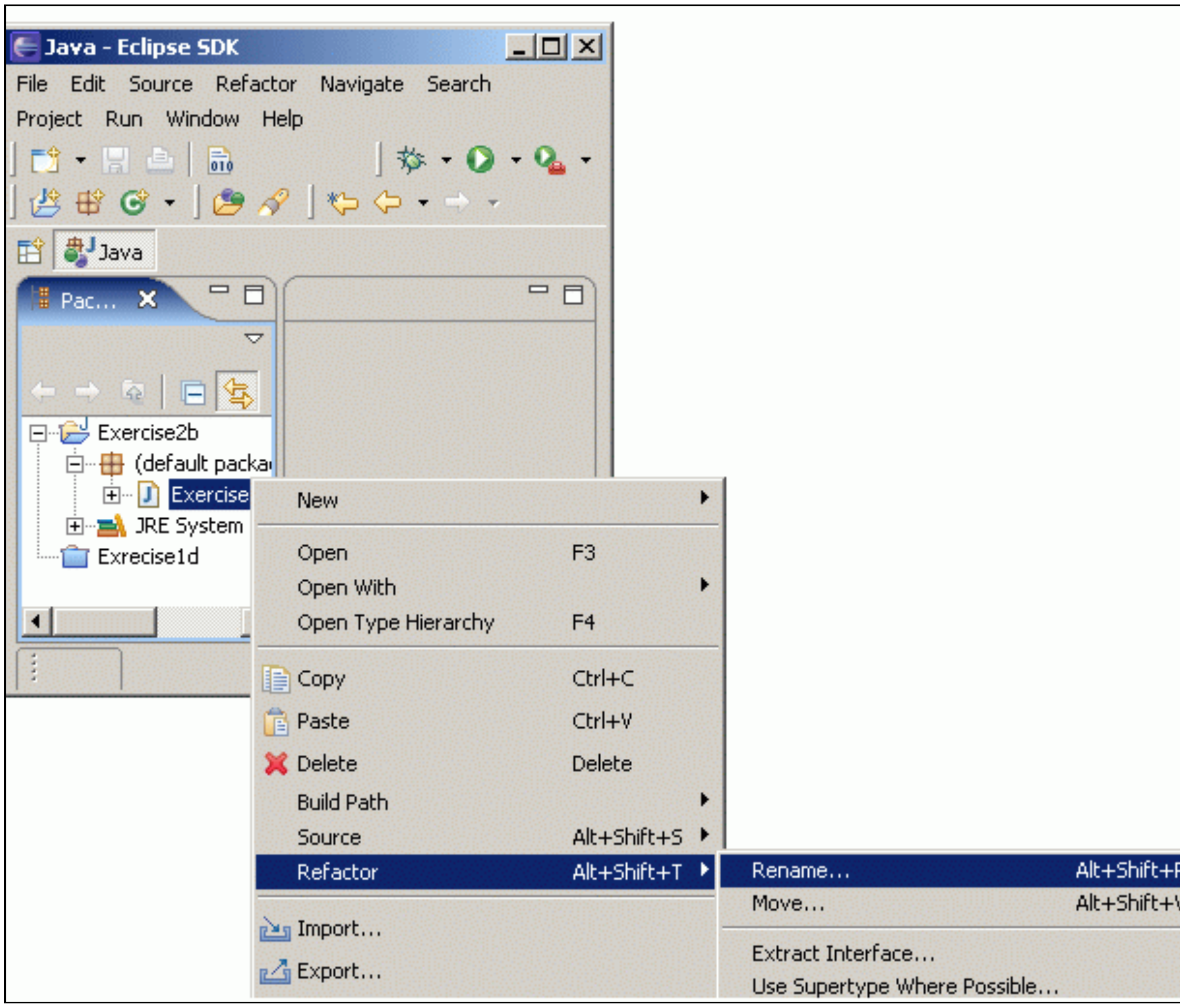
It applies to an older version of 114, so although specific file names and projects no longer apply, the concepts still hold.

You will now change *Exercise1d.java* in the *Exercise2b* project to *Exercise2b.java* :

1. Make sure *Exercise1d.java* is visible (the class you just copied in the Copying a Class tutorial). If it is not, click on the arrow beside the *Exercise2b* project, click the arrow beside *src* (not shown in the screenshot), then click on the arrow beside *(default package)* .


Java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Package ...
Exercise1d
Exercise2b
(default package)
Exercise1d.java
JRE System Library
Console
Exercis...cise2b

2. Right-click *Exercise1d.java* , and highlight *Refactor* in the context menu that appears, then choose *Rename*... from the submenu that appears.



A *Rename Compilation Unit* pop-up window will appear.

3. Change "Exercise1d" to "Exercise2b" beside the *New name:* field, then click the *Finish* button.

Not only have you renamed the file, but the class name and any other occurrences of *Exercise1d* in the *Exercise2b* project have been changed to *Exercise2b* .

Changing an Applet's Width and Height

1. Compile the Applet project.
2. Select the Applet project. Ensure you've run it at least once.
3. Choose *Run Configurations*... from the *Run* menu.
4. Change Width and Height appropriately in the *Parameters* tab, and click *Run* .

Handling Infinite Loops

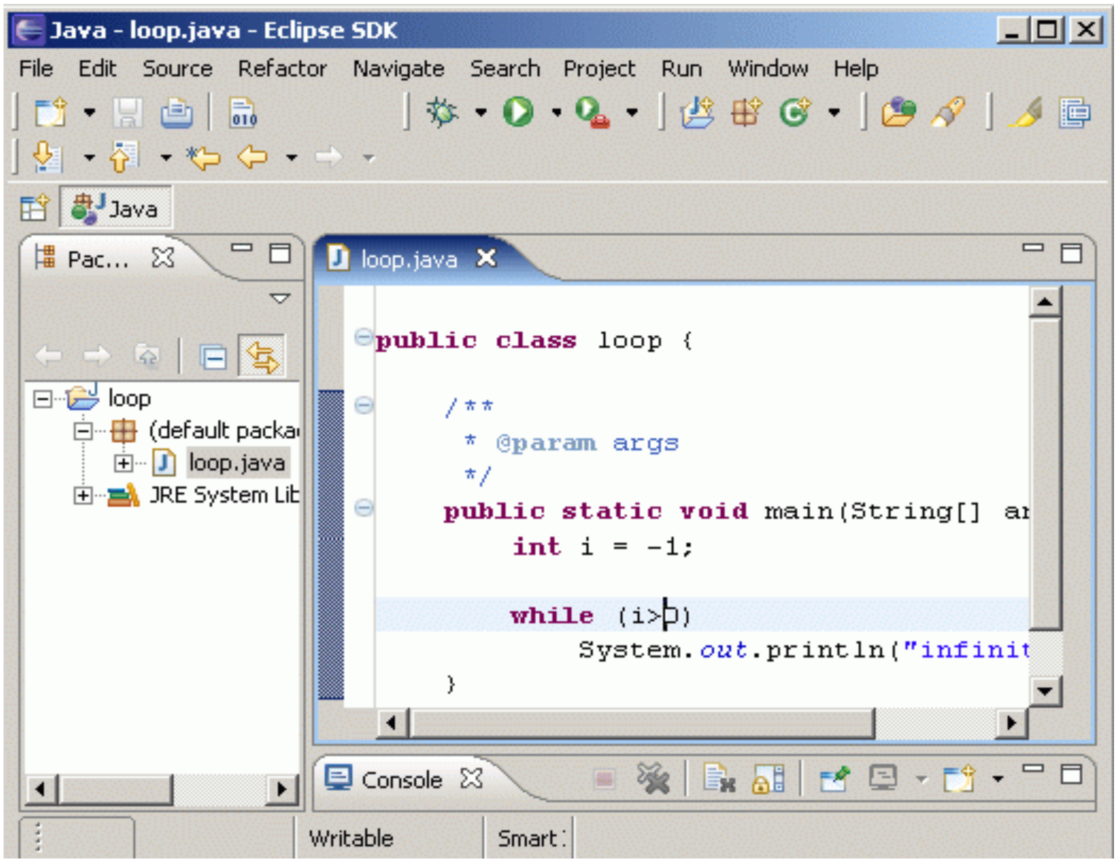
You first need to recognize that your program has an infinite loop. This isn't obvious when new to Eclipse, but once you know what to look for, and get in the habit of checking for infinite loops, they are easy to detect.

Once you have diagnosed an infinite loop, you must end your program by terminating the infinite loop, and then should attempt to fix your code.

A related problem occurs when you have many programs running at the same time which slows Eclipse down. This often happens because you have had an infinite loop which you did not previously detect. To rectify this, you need to terminate these previous processes .

Recognizing an Infinite Loop

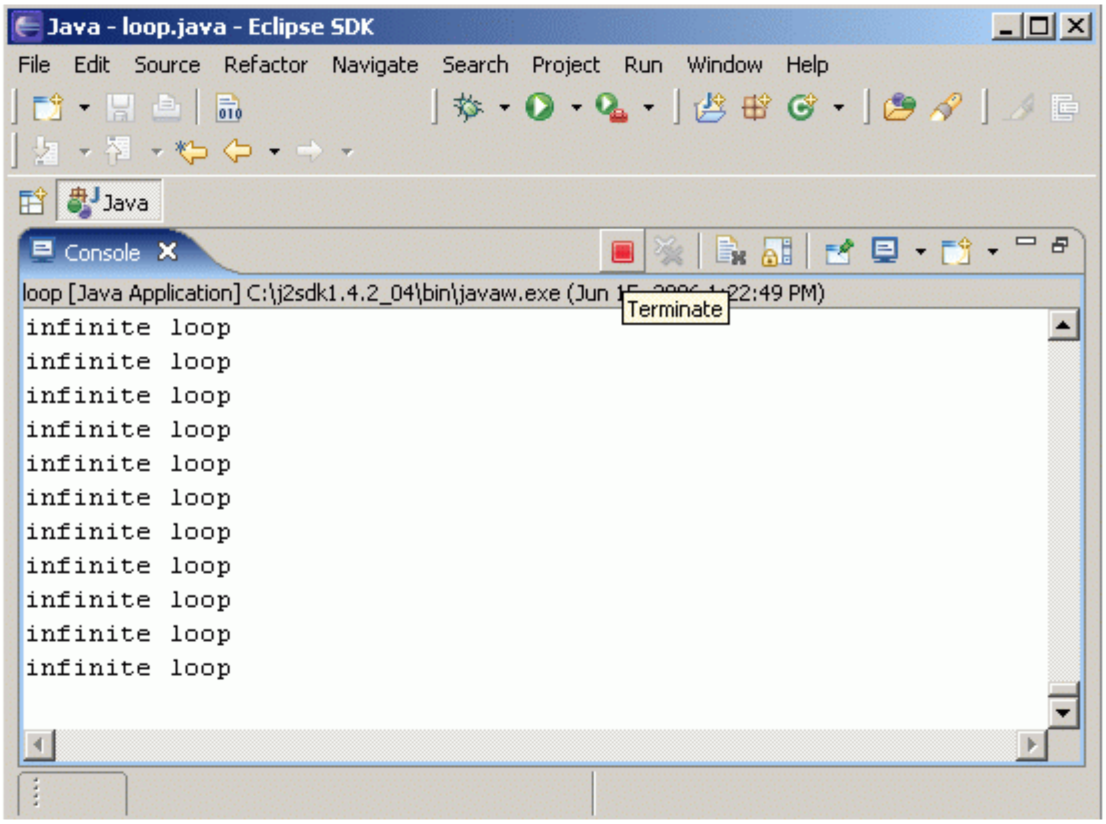
In Eclipse, a square button in the Console View at the bottom of the Eclipse window will indicate the presence/absence of an infinite loop. If your program has terminated, the button will remain greyed out. (This is not quite true.)


Java - loop.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Pac...
loop.java
loop
(default packa
loop.java
JRE System Lib
Console
Writable
Smart

```
public class loop {

    /**
     * @param args
     */
    public static void main(String[] ar
        int i = -1;

        while (i>0)
            System.out.println("infinit
    }
```

There is no infinite loop to terminate in this case.

But, if your program is looping endlessly, the button will become a RED SQUARE :


Java - loop.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Console
Terminate

loop [Java Application] C:\j2sdk1.4.2_04\bin\javaw.exe (Jun 15, 2006 1:22:49 PM)

```
infinite loop
infinite loop
infinite loop
infinite loop
infinite loop
infinite loop
infinite loop
infinite loop
infinite loop
infinite loop
infinite loop
```

You can now terminate that infinite loop.

Get into the habit of checking for infinite loops after each program, especially if your program contains at least one loop. Note also that an infinite loop is quite obvious if your program produces output within the loop. The program above outputs "infinite loop" infinitely.

The following is likely **not necessary** to read, but does describe things more accurately.

The square button is red whenever the program **executes**, not only for infinite loops. For example, when your program is supposed to run for a long time, or when you are using the Eclipse Debugger, the square button is red, but you have no infinite loop.

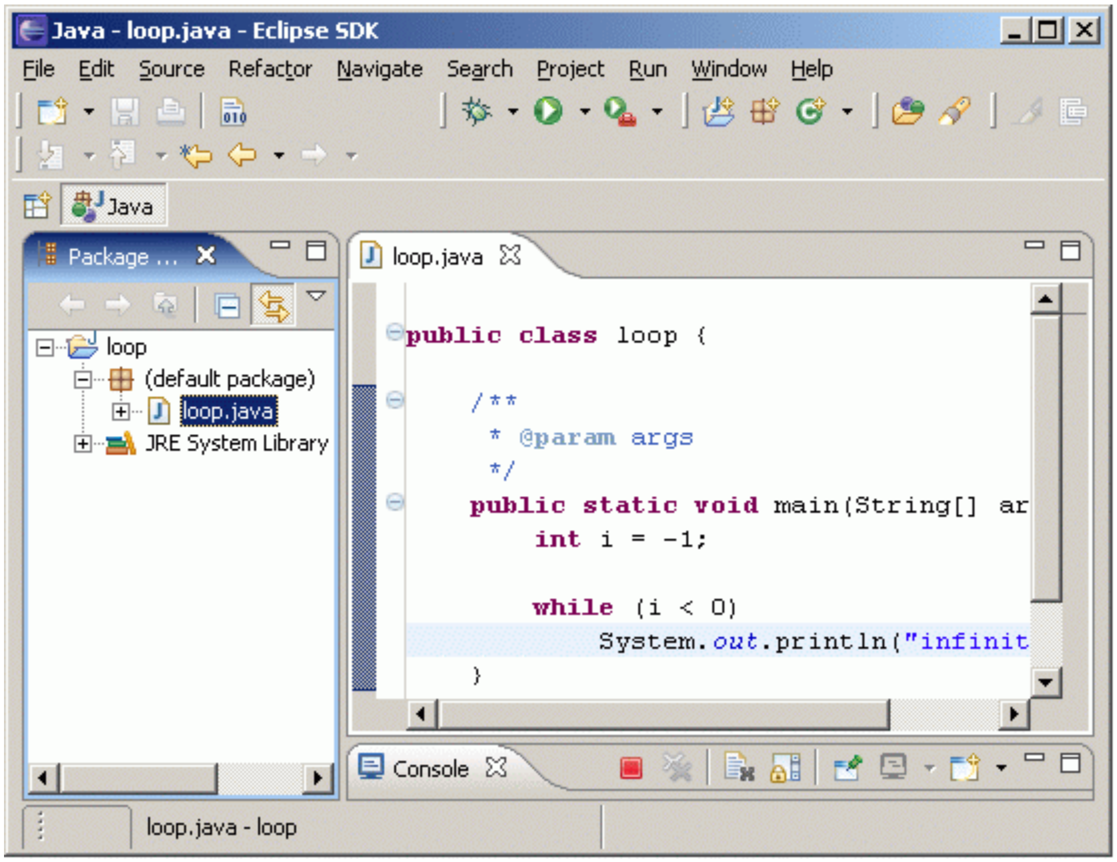
So, although the previous description suffices in most cases, realize you have an infinite loop only when the square button remains red **longer than expected**.

Terminating an Infinite Loop

Once you have recognized that an infinite loop is running, terminate it by doing the following:

- Go to the Console View at the bottom of the Eclipse window
- Click on the RED SQUARE button shown below, sometimes you may have to click more than once.

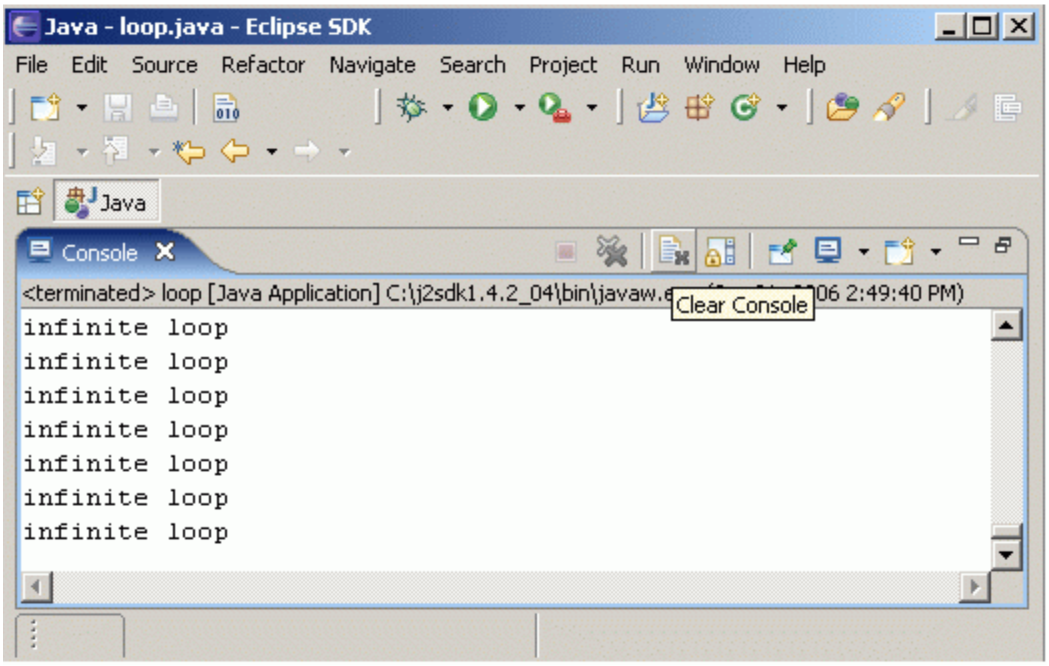
The infinite loop will now end.


Java - loop.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Package ...
loop
(default package)
loop.java
JRE System Library
loop.java
Console
loop.java - loop

```
public class loop {

    /**
     * @param args
     */
    public static void main(String[] ar
        int i = -1;

        while (i < 0)
            System.out.println("infinit
    }
```

The RED SQUARE button can also be used to terminate a program if it is waiting for user input.

After terminating the infinite loop, you may also need to erase buffered output from the Console by clicking the Clear Console button.


Java - loop.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Console
Clear Console

```
<terminated> loop [Java Application] C:\j2sdk1.4.2_04\bin\javaw.e ... 06 2:49:40 PM)
infinite loop
infinite loop
infinite loop
infinite loop
infinite loop
infinite loop
infinite loop
```

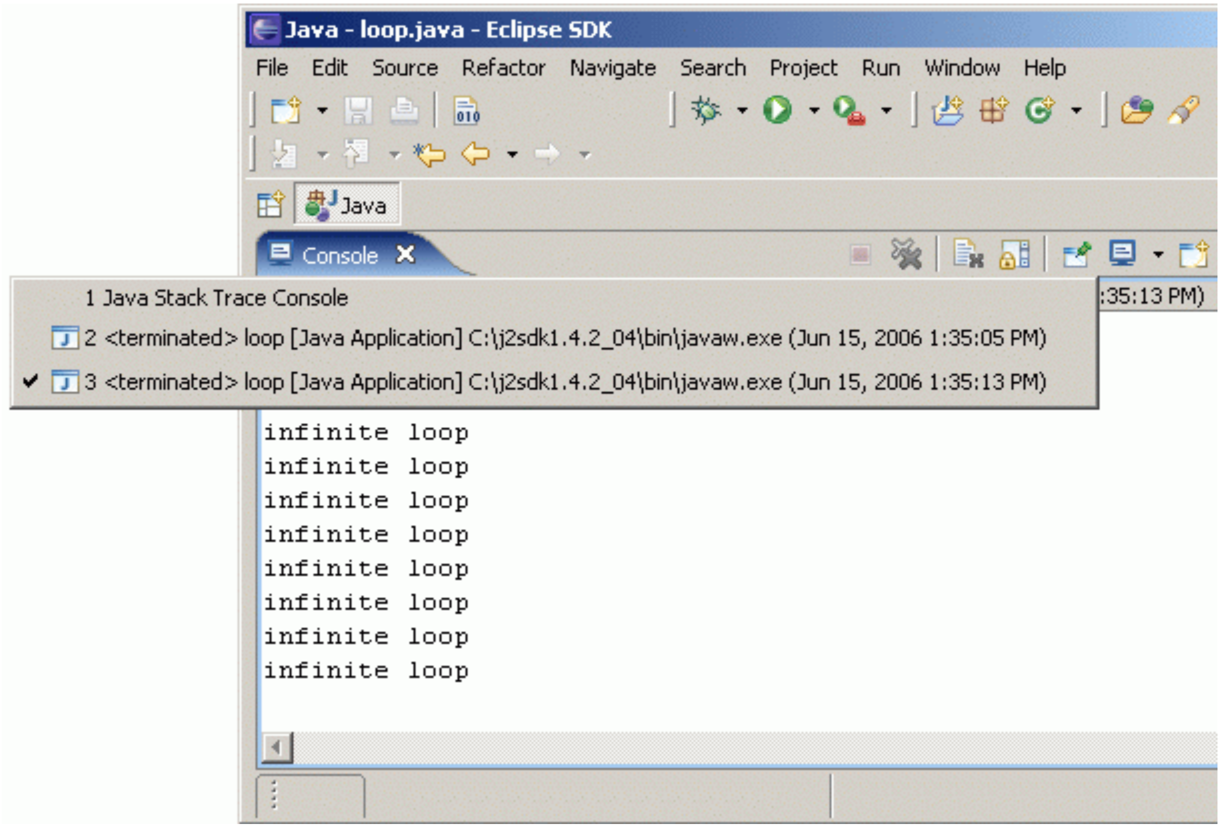
Ending Previous Programs (Often Solves Eclipse Acting Slowly)

Eclipse will slow down if you have multiple Java programs running. This often happens if you forget to terminate previous infinite loops, or previous programs that are waiting for input.

Never close Eclipse while it is running one or more programs. This will cause more problems than it will solve.

Instead, you can prevent Eclipse from slowing down by always detecting and then terminating infinite loops (or programs waiting for user input).

Or, if you forget to do this, go to the Console View at the bottom of the Eclipse window, and click on the small down arrow beside the Display Selected Console icon:


Java - loop.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Console

```
1 Java Stack Trace Console
2 <terminated> loop [Java Application] C:\j2sdk1.4.2_04\bin\javaw.exe (Jun 15, 2006 1:35:05 PM)
✓ 3 <terminated> loop [Java Application] C:\j2sdk1.4.2_04\bin\javaw.exe (Jun 15, 2006 1:35:13 PM)
```

```
infinite loop
infinite loop
infinite loop
infinite loop
infinite loop
infinite loop
infinite loop
infinite loop
```

In the menu that appears, any process that **does not** have the phrase <*terminated*> beside it needs to be terminated (the third one in the above example, whereas the 2nd one has terminated properly). Click on each process and terminate it as you terminated infinite loops.

Importing an eclipse project into your workspace

(This assumes that you have already copied your project(s) into your workspace directory)

1. In Eclipse, go to the *File* menu, and choose *Import*....
2. In the Import sub-window, click the arrow beside General. It expands.
3. Under General, choose *Existing Projects into Workspace*.
4. Click the *Next* > button.
5. Click *Browse*..., then navigate to your workspace directory

6. Click OK
7. Click the *Finish* button.

(There are other, likely better ways to do this such as copying the project directories by checking the 'copy projects into workspace' checkbox while importing.)

Reconfiguring Eclipse/Fixing Eclipse Problems In-lab

If Eclipse is not configured properly, or you get some "weird" problems with Eclipse, you may need to reinstall your Eclipse workspace.Some examples of "weird" problems follow:

- Being unable to create a project, even though you are doing everything correctly.
- Not being able to open projects.
- Noticing that other normally present Eclipse functionality is missing.
- Other specific problems and solutions

Ask your TA first, as there may be an easier and less potentially dangerous fix than the following.

1. **Ask a TA for help**; it's not recommended you do this on your own
2. Save all your work in Eclipse, then close Eclipse
3. **Back-up your workspace** This means make a copy of the entire workspace directory.
4. If you haven't done so already back-up your workspace (yeah, it's important)
5. Delete your original workspace which, of course, you've already backed-up
6. Download workspace.zip to your home directory.
7. Navigate to your home directory, unzip workspace.zip by double-clicking on workspace.zip and extracting the files. (We're hoping you will quickly see how to extract files. If not, please ask a TA.) A **new workspace** folder will be created.
8. Copy over any Eclipse project directories you want to use from your backed-up workspace into the new workspace directory
9. Then import each project into Eclipse.

Potential Problems / Fixes

- You may get an error similar to the following in the "Problems" view:

Description	Resource	Path	Location	Type
Access restriction: The field in from the type Keyboard is not accessible due to restriction on required library C:\Program Files\Java\jdk1.6.0_31\jre\lib \ext\UofAC114.jar	test3a.java	/test4/src	line 14	Java Problem

  You need to configure Eclipse like in the lab. There are more details in the Keyboard section.

- If you get "Exception in thread "main" java.lang.NoClassDefFoundError" (in the Problems View):

  You may need to build your project first.

Other Handy Features

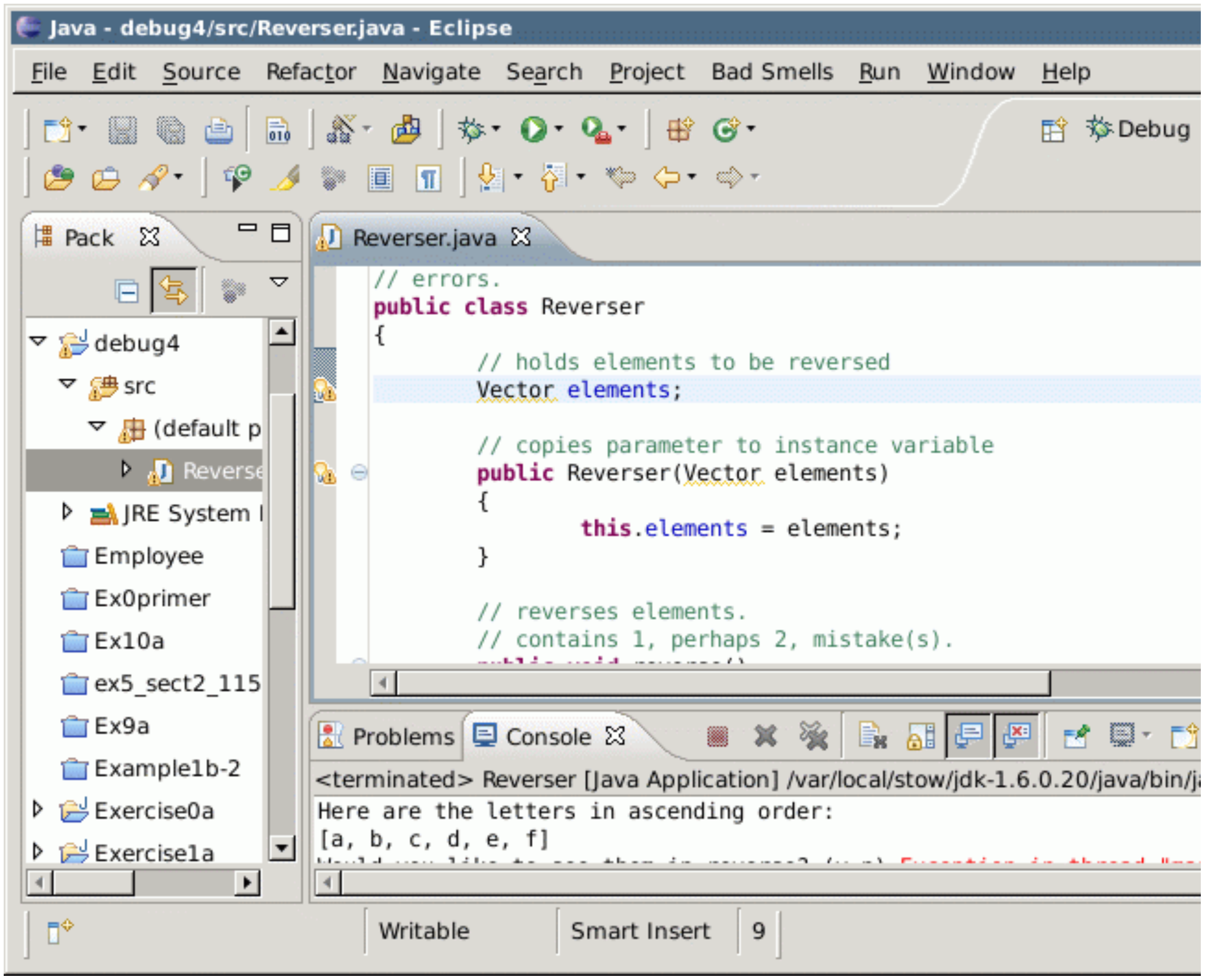
Eclipse has a number of useful tools that you may find handy while using Eclipse.

Refactoring Code

Refactoring is a useful feature if you ever decide to move or rename a variable or class. This guide is more focused on renaming variables. To refactor classes, see Renaming a class.

Note that several other types of code refactoring are possible besides renaming, but those will not be explored in depth here. Feel free to experiement with the available options though.

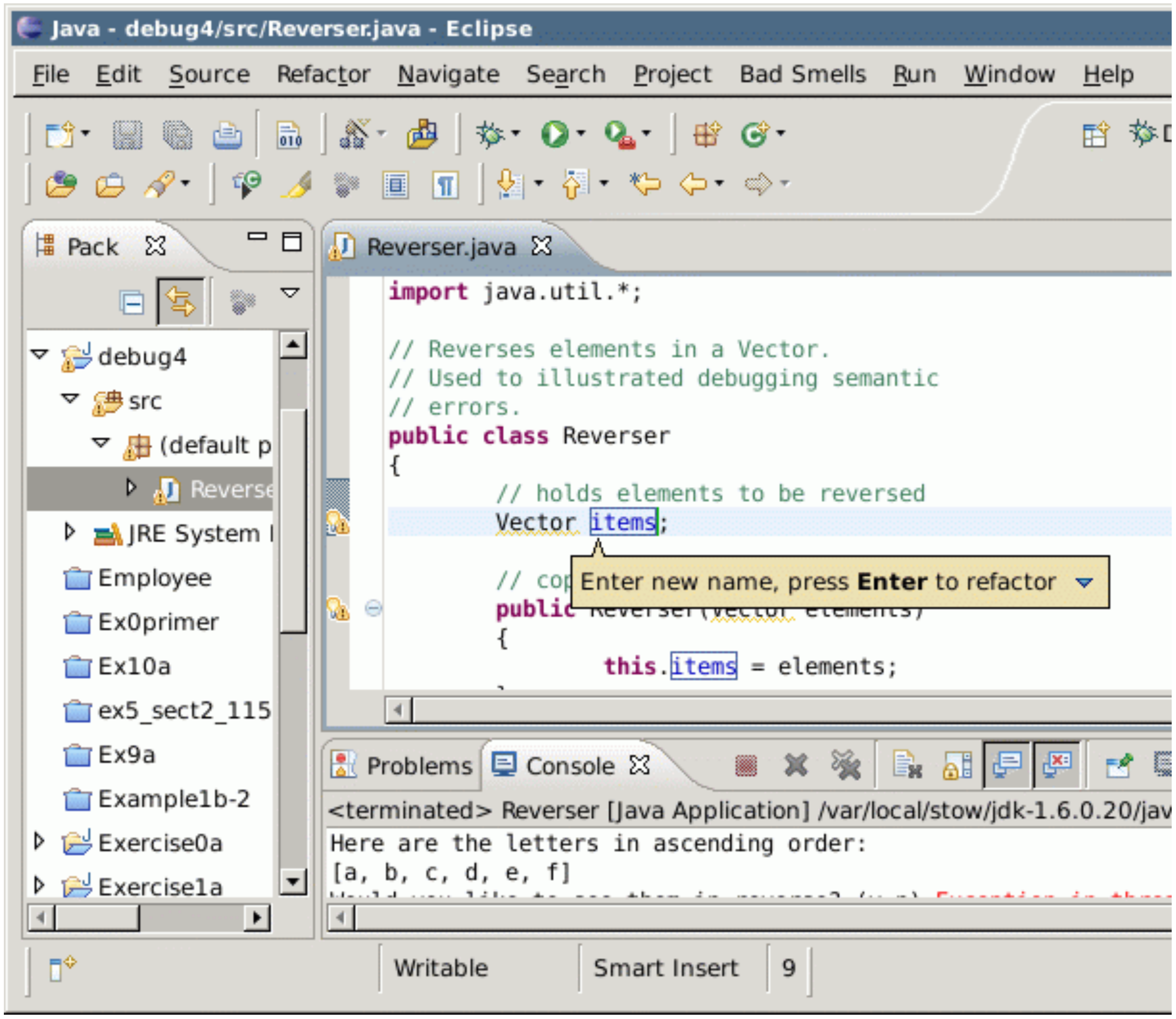
Imagine you have the code below, and you want to rename the `elements` variable to `items`:



```
// errors.
public class Reverser
{
        // holds elements to be reversed
        Vector elements;

        // copies parameter to instance variable
        public Reverser(Vector elements)
        {
                this.elements = elements;
        }

        // reverses elements.
        // contains 1, perhaps 2, mistake(s).
```

```
<terminated> Reverser [Java Application] /var/local/stow/jdk-1.6.0.20/java/bin/j
Here are the letters in ascending order:
[a, b, c, d, e, f]
```

1. Click on the word `elements`, then select "Rename" from the "Refactor" menu.
2. A colored box will appear around the variable name. As you edit the variable name, all other instances of that variable will also change in the code. This means that not only will instances of Reverser.elements be changed in this file, but any other references to this variable in *other* classes in the project will be changed as well.



```
import java.util.*;

// Reverses elements in a Vector.
// Used to illustrated debugging semantic
// errors.
public class Reverser
{
        // holds elements to be reversed
        Vector items;

        // co
        public Reverser(Vector elements)
        {
                this.items = elements;
```

```
<terminated> Reverser [Java Application] /var/local/stow/jdk-1.6.0.20/jav
Here are the letters in ascending order:
[a, b, c, d, e, f]
```

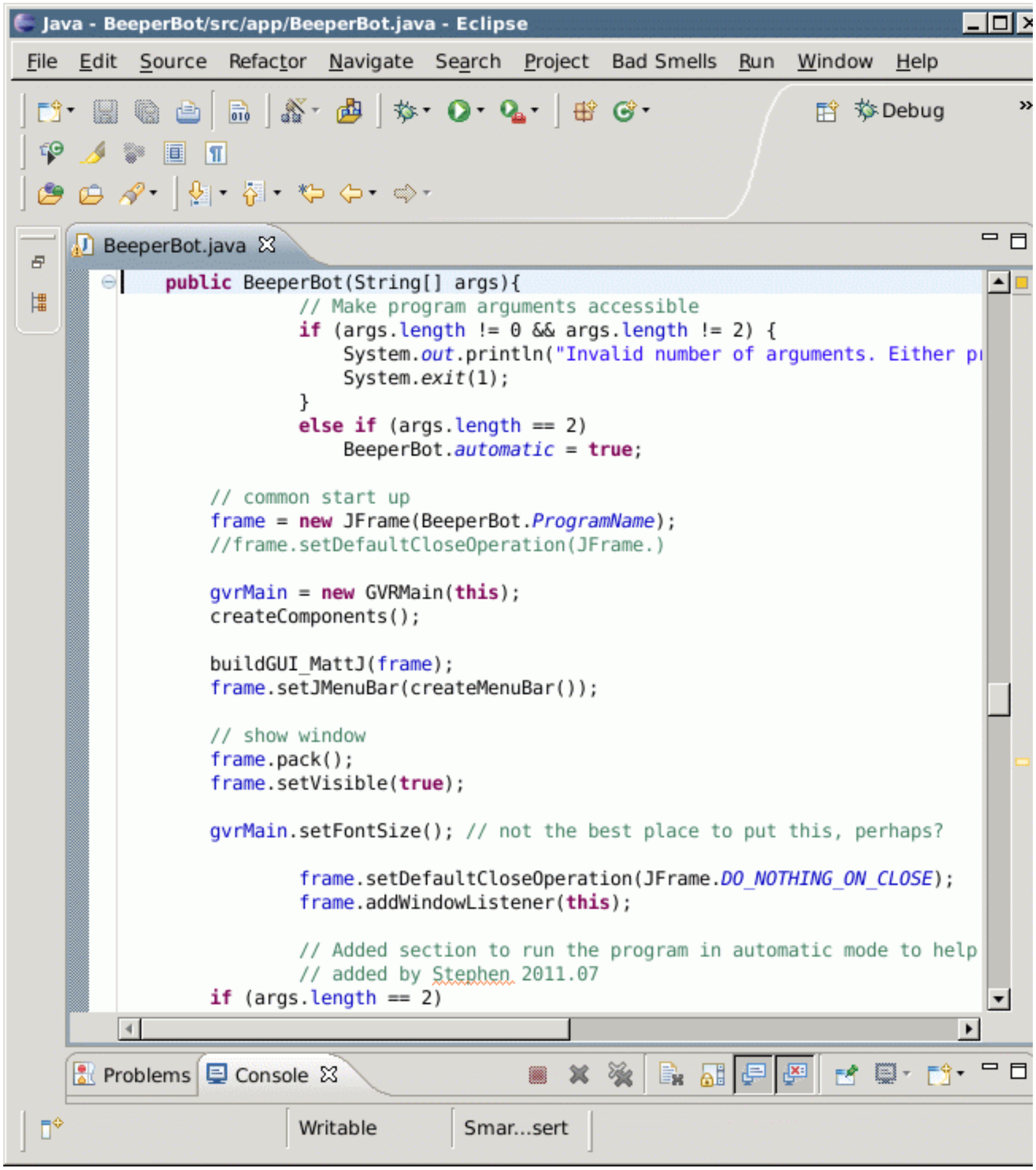
3. Press Enter to finish refactoring.

Formatting Code

Eclipse can also reformat code: it will correct indentation, align trailing comments, and in general make code more readable.

Note: The format feature should not be exclusively relied upon to clean up your code. In general, you should try to produce cleanly indented code on your own, as it is simply proper form, and because not all IDEs you may encounter have this feature.

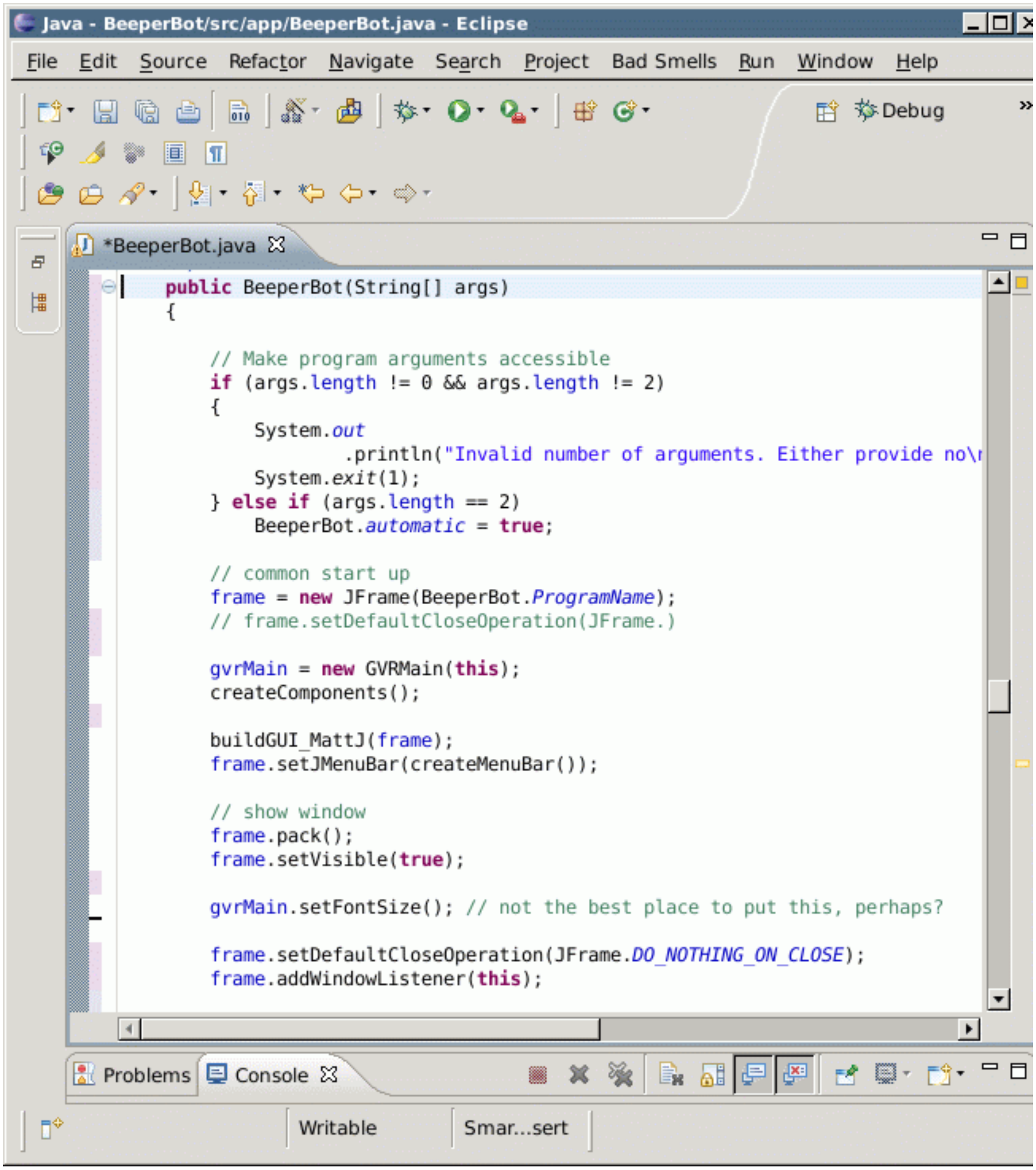
Take the code below for example:


Java - BeeperBot/src/app/BeeperBot.java - Eclipse
File Edit Source Refactor Navigate Search Project Bad Smells Run Window Help
Debug
BeeperBot.java
Problems
Console
Writable
Smar...sert

```
    public BeeperBot(String[] args){
                // Make program arguments accessible
                if (args.length != 0 && args.length != 2) {
                    System.out.println("Invalid number of arguments. Either pi
                    System.exit(1);
                }
                else if (args.length == 2)
                    BeeperBot.automatic = true;

        // common start up
        frame = new JFrame(BeeperBot.ProgramName);
        //frame.setDefaultCloseOperation(JFrame.)

        gvrMain = new GVRMain(this);
        createComponents();

        buildGUI_MattJ(frame);
        frame.setJMenuBar(createMenuBar());

        // show window
        frame.pack();
        frame.setVisible(true);

        gvrMain.setFontSize(); // not the best place to put this, perhaps?

                frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
                frame.addWindowListener(this);

                // Added section to run the program in automatic mode to help
                // added by Stephen 2011.07
        if (args.length == 2)
```

Glancing at the code, you will notice that the indentation is rather messy.

1. Highlight the code you want to reformat.
2. From the Source menu, select Format (or type *Shift+Ctrl+F*)


Java - BeeperBot/src/app/BeeperBot.java - Eclipse
File Edit Source Refactor Navigate Search Project Bad Smells Run Window Help
Debug
*BeeperBot.java
Problems Console
Writable Smar...sert

```
public BeeperBot(String[] args)
{

    // Make program arguments accessible
    if (args.length != 0 && args.length != 2)
    {
        System.out
                .println("Invalid number of arguments. Either provide no\r
        System.exit(1);
    } else if (args.length == 2)
        BeeperBot.automatic = true;

    // common start up
    frame = new JFrame(BeeperBot.ProgramName);
    // frame.setDefaultCloseOperation(JFrame.)

    gvrMain = new GVRMain(this);
    createComponents();

    buildGUI_MattJ(frame);
    frame.setJMenuBar(createMenuBar());

    // show window
    frame.pack();
    frame.setVisible(true);

    gvrMain.setFontSize(); // not the best place to put this, perhaps?

    frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);
    frame.addWindowListener(this);
```

After reformatting is finished, the code is correctly indented, and you can see that Eclipse attempted to shorten a very long `System.out.println` line by breaking it into two.

You can also reformat an entire file by having nothing selected when you click the Format menu item.

Installing Eclipse at Home

To get Eclipse to work as in-lab, you will need to install Java, install Eclipse and then configure Eclipse. This section shows you roughly how to do this on Windows. See below for a guide to installing Eclipse on a Mac. Note the following:

- This guide is geared towards installing the lab versions for Java and Eclipse on a Windows Vista 32-bit computer (it is likely similar on Windows 7). It was done use Firefox.
- Basic familiarity with Windows, Firefox (or Internet Explorer) and WinZip (or 7-zip or the Windows built-in extractor) are assumed.
- If a link is mentioned, and you cannot find that link, search for that link (i.e., using Ctrl-F in Firefox.)
- This guide may not be fully applicable since the Sun site (where to download Java) and the Eclipse site are in constant flux.
- There are many other ways to install (and configure) Java and Eclipse.
- If you have problems, please ask your TA; if they can help, they will. Note that you are ultimately responsible for solving problems on your home computer, though.
- Installing Eclipse and/or Java should not harm your computer, yet Computing Science staff are not responsible for any problems that occur as a result of following these instructions.
- Eclipse does not need to be installed at home; you can work in-lab, or can program in Java at home using software other than Eclipse.

Downloading and Installing Java

Ensure you read over the above notes before installing Java. The following is a guide for installing the Java Development Kit (JDK) 1.6.0_31 , which is similar to the version installed in the labs.

1. Go to the 1.6.0_31 download page.
2. Read and accept the license agreement.
3. Right-click on *jdk-6u31-windows-i586.exe* link, choose *Save Link As...*, and save the file. (On Windows 7, using IE, it'll be *Save target as*.)
   - Note this file is for 32-bit windows.
   - Save it in an appropriate place, such as your Desktop.
4. Go to where you saved that file (e.g., your Desktop), and double click on *jdk-6u31-windows-i586.exe*, and follow the instructions.
   - You may need to enter an Administrator password before running the .exe file.
   - If the installation fails, try disabling installation of certain unnecessary components, such as "source code" - this may fix the installation issues you are having. This has been an issue in rare cases on Windows 7.
   - Note a later step, configuring Eclipse, assumes you chose the default location C:\Program Files\Java\jdk1.6.0_31\ [or similar, likely C:\Program Files (x86)\Java\jdk1.6.0_31, if on a 64-bit Windows 7 system]
   - Later during the install, you will be prompted to install into C:\Program Files\Java\jre6. Accept this, but note this is a different part of the installation which will not be referenced in this tutorial.

See below for a guide to installing Eclipse on a Mac. Note you do not need to install Java on a Mac.

You should now install Eclipse.

Downloading and Installing Eclipse

Ensure you read over the above notes before installing Eclipse. The following is a guide for installing Eclipse 3.6.2, which is the same version as in the labs.

1. Save eclipse-java-helios-SR2-win32.zip to your computer.
2. Unzip it to your C drive (If your system does not have a built-in ability to unzip .zip files, you can use 7-zip.)
   - You may need to save it to your Desktop, then move it to your C drive, entering an Administrator password.
   - You will likely need to explictly specify C:\ as the location of the extract.
   - Note the next step, configuring Eclipse, assumes you have a resulting C:\eclipse folder.

You should now configure Eclipse.

See below for a guide to installing and configuring Eclipse on a Mac.

Configuring Eclipse Like in the Lab

Ensure you read over the above notes (this applies to a Windows machine ... see the bottom of this section for a link to configuring Eclipse on a Mac), have installed Java, and have installed Eclipse before configuring Eclipse. The following is a guide for configuring Eclipse 3.6.2 as it is configured in the labs.

1. If installing Eclipse, download workspace.zip, and unzip it in your C drive, resulting in `C:\workspace`.
   - You may need to save it to your Desktop, then move it to your C drive, entering an Administrator password.
   - If your system does not have a built-in ability to unzip .zip files, you can use 7-zip
   - You will likely need to explictly specify C:\ as the location of the extract.
   - In C:\workspace is a file WORKSPACE_3.6.2_SETTINGS. It's a text file containing all modified settings.
2. Create a shortcut to `C:\eclipse\eclipse.exe`.

   Note This assumes you installed Eclipse as described above. If you have not, install things again, but this time in the specified locations, or adjust appropriately during the following steps.

   1. Right click on `C:\eclipse\eclipse.exe`.
   2. Choose *Send To -> Desktop*.

   See this Vista tutorial on shortcuts for more details.

3. Modify the shortcut's target so it has the following command

```
C:\eclipse\eclipse.exe -vm "C:\Program Files\Java\jdk1.6.0_31\bin\javaw.e
```

You can now double-click on the shortcut to run Eclipse.

Note:

- *-vm C:\Program Files\Java\jdk1.6.0_31\bin\javaw.exe* is the location of your Java installation.

  Note that if you are using 64-Bit Windows, Java may be installed in *C:\Program Files* ***(x86)****\Java\jdk1.6.0_31\bin\javaw.exe*

  If you get a message similar to "A Java Runtime Environment (JRE) or java Development Kit (JDK) must be available in order to run Eclipse. No Java virtual machine was found after searching the following location: C:\Program Files\Java \jdk1.6.0_31\bin\javaw.exe" you may need to change this path.

- *C:\workspace* is the location of your eclipse projects containing your .java files.
- *-vmargs -Dfile.encoding=utf-8* was in the past needed for 115 projects.

See below for a guide to installing Eclipse on a Mac.. It also shows you how to configure it roughly like in the lab.

Getting Keyboard to Work at Home

(This applies to Windows ... see the bottom of this section for a link to getting Keyboard to work on a Mac.)

Close down Eclipse, first.

Then, to get Keyboard.java and RandomInt.java working at home, you must install UofAC114.jar . The easiest and best way to do this is as follows:

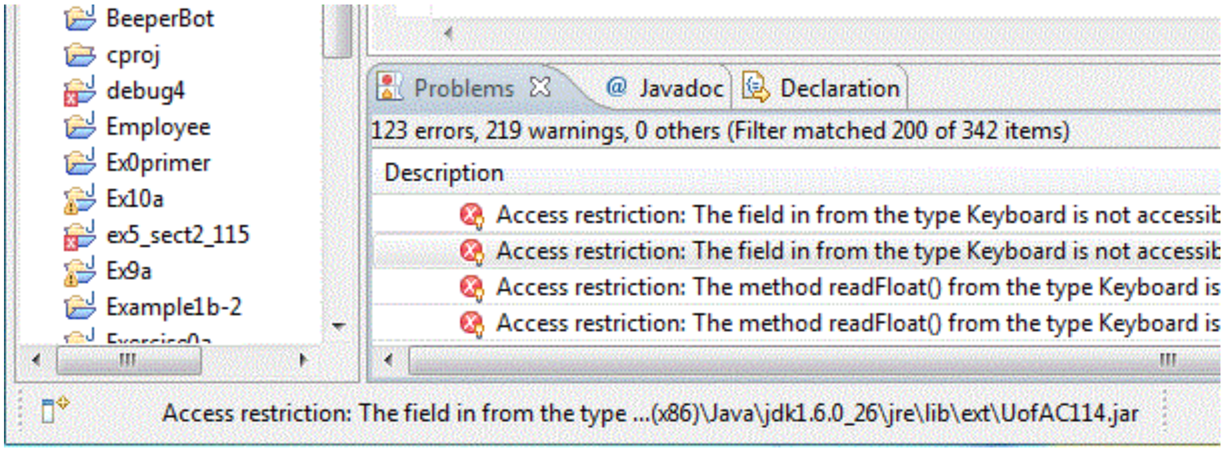
1. Right click on this UofAC114.jar link and click *Save Link As ...* . An *Enter name of file to save to...* window appears. (Applies to Firefox on Windows Vista.)

2. Save it in the *lib\ext* directory of your java installation. If you installed Java as described above, then it'll be similar to

   ```
   C:\Program Files\Java\jdk1.6.0_31\jre\lib\ext
   ```

   - If it does not allow you to save this file, try saving it to your Desktop, then moving it to the ext directory, entering your Administrator user's password if and when prompted.
   - If on a 64-bit system, it's likely you must instead use this path: `C:\Program Files` **`(x86)`**`\Java\jdk1.6.0_31\jre\lib\ext`
3. Start up Eclipse, and now any project will have Keyboard available.

- If you have installed Keyboard as above and you are not using the provided workspace directory, your programs that use Keyboard may not compile, with the errors displayed below.



123 errors, 219 warnings, 0 others (Filter matched 200 of 342 items)

Access restriction: The field in from the type Keyboard is not accessib
Access restriction: The field in from the type Keyboard is not accessib
Access restriction: The method readFloat() from the type Keyboard is
Access restriction: The method readFloat() from the type Keyboard is

Access restriction: The field in from the type ...(x86)\Java\jdk1.6.0_26\jre\lib\ext\UofAC114.jar

To fix this problem, you must configure Eclipse like in the lab.

(This example applies to a Windows machine, but it is similar to other OSs and in-lab.)

See below for how to get Keyboard working for a Mac

That be it. If it doesn't work, ensure you've installed Java, installed Eclipse and configured Eclipse properly, as specified above. And, if needed add the Keyboard.java or RandomInt.java files directly to each project where they must be used. These files must be added under the *(default package)* , or to the project itself, but not to the *JRE System Library* .

For more advanced courses, you may need to install Eclipse plug-ins.

Installing and Configuring Eclipse at Home on a Mac

You may wish to see some notes about the installation on a Mac.

1. In your browser, go to http://www.eclipse.org/
2. Select Downloads.
3. Select Older Versions (easiest to search for it; it's a bit hard to spot).
4. Select Eclipse Helios SR2 Packages (v 3.6.2)
5. Select the Eclipse IDE for Java Developers.
6. On the right side of the page, click Mac OS X(Cocoa 32).
    - For a 64-bit Mac, click Mac OS X(Cocoa 64)
7. Click the green download arrow. It should download eclipse-java-helios-SR2-macosx-cocoa.tar.gz
    - For a 64-bit Mac, it should download eclipse-java-helios-SR2-macosx-cocoa-x86_64.tar.gz
8. When the download is complete, move the archive into the appropriate folder and unzip it. (Likely unzip by double-clicking.) Eclipse is now installed.
    - Put it in the Applications folder, if you're not certain which folder is appropriate.
9. **configuring Eclipse like in the lab:** Download workspace.zip. Move it to the Documents folder.

    If it didn't automatically unzip, double-click to unzip it. A workspace directory will be created.

This contains the configuration settings for Eclipse, and will also contain your .java files.

10. Go into the eclipse folder (created when you unzipped it), and run Eclipse.
    - If you put it in the Applications folder, go to Applications in the dock, go into the Eclipse folder, and single click Eclipse (or Eclipse.app).
    - If you did not put Eclipse in the Applications folder, then in Finder go to the Eclipse folder, and double-click Eclipse (you may see it as Eclipse.app).

    If you get a message similar to *Eclipse is an application downloaded from the Internet. Are you sure you want to open it?*, click Open.

11. When prompted, select the workspace (/Users/*your_Mac_user_name*/Documents/workspace), and choose *use this as the default and do not ask again*.

Now move onto the next section where you specify the Java version.

Specifying Java Version in Eclipse

Note you may only have one choice, in which case you cannot specify another version. It should not in this case be needed, though.

1. At the top of the Eclipse window select the Eclipse menu.
2. Select Preferences... .
3. In the left column of the window that appears, expand Java.
4. Select Installed JREs.
5. From the menu that appears choose JVM 1.6 and then press Okay.

Eclipse is now using the Mac version of Java closest to the Java used in-lab.

Notes on Mac Installation

- Done on Mac OS X 10.5.8, with a Power PC G4 (which is 32 bit).
    - Also tested briefly on 10.7.1, 64 bit.
- It was assumed the Mac supported both the Cocoa and Carbon versions of Eclipse, but Cocoa was chosen because it was newer.
- When choosing the Java version, 1.6 was arbitrarily chosen over 1.6.0.
- Eclipse should work similarly to the lab, but there will be at least a few differences. For example, short cut keys will be different.
- The version of Java is not exactly the same as the lab (i.e., 1.6 was used instead of 1.6.0_31 ... using 1.6.0_31 on a Mac is likely not possible), and the configuration settings are not exactly the same either (no command line arguments were given). You can try creating a shortcut, which on a Mac is called an alias. Using the eclipse.ini file may also work.
- It is unnecessary to install Java, as Macs come with Java already installed.

Getting Keyboard to Work at Home (Mac)

Ensure you've first followed the installation instructions precisely.

1. Before creating the Eclipse project, close Eclipse.
2. In Finder, go to the Macintosh HD (in Finder, left pane under Devices. In Finder, you may first

instead need to go to the Go menu, and choose Computer. It also may display as the name you chose for your computer, instead of Macintosh HD.)
3. Choose Library -> Java -> Extensions
4. Download UofAC114.jar and save it in Extensions.
5. Start up Eclipse, and now any project will have Keyboard available.

If that didn't work, so for example code with Keyboard is still not compiling, first ensure you put UofAC114.jar in the proper path, as specified above ... it must **not** be, for example, if your user directory's Library/Java/ext directory.

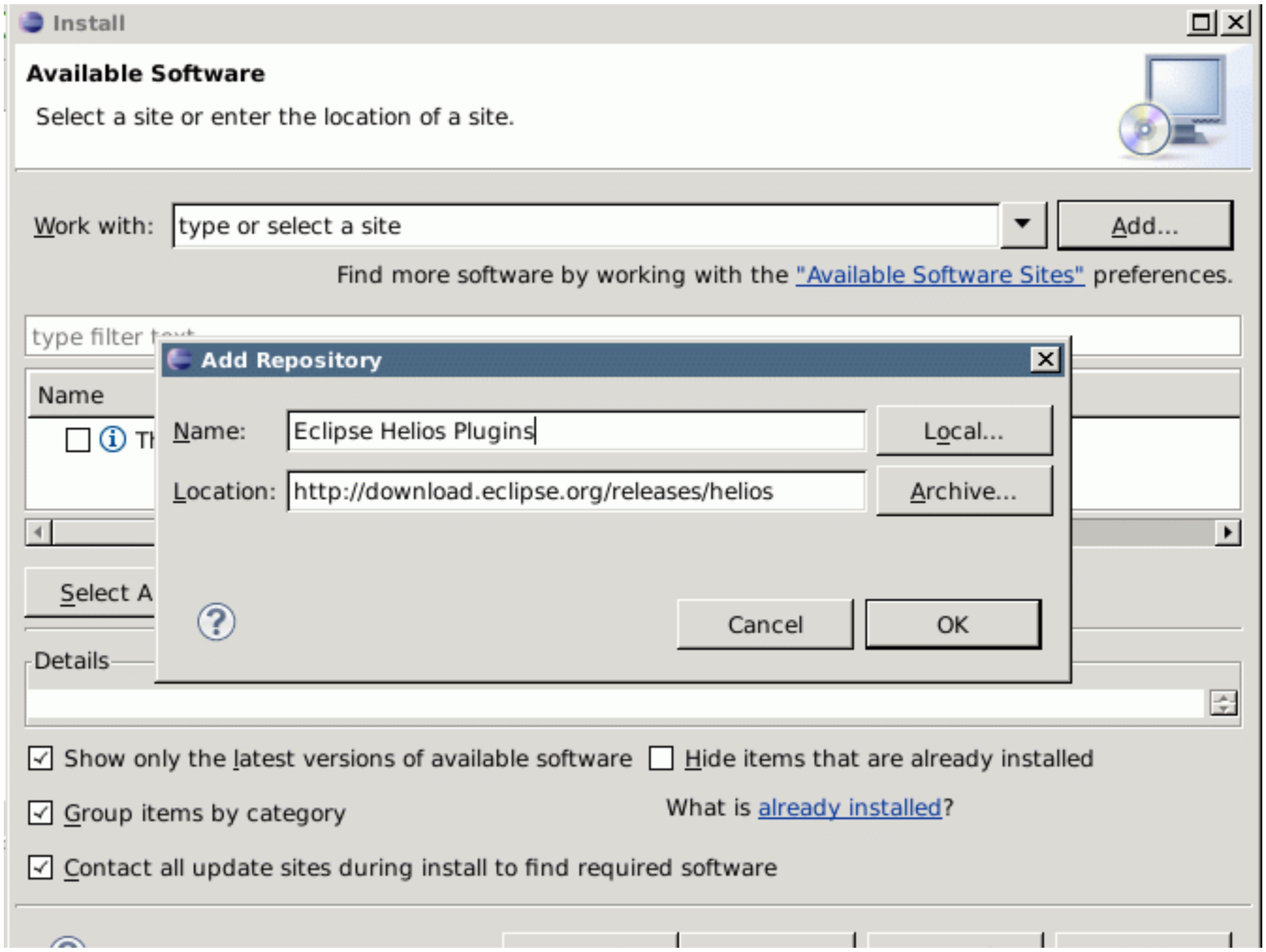
If you get an Access restriction error, you likely did not configure Eclipse properly.

You can also try some alternatives.

For more advanced courses, you may need to install Eclipse plug-ins.

Installing Eclipse Plug-Ins

See Installed Plugins for a list of plugins installed on ugrad computers.


Install
Available Software
Select a site or enter the location of a site.
Work with:
type or select a site
Add...
Find more software by working with the "Available Software Sites" preferences.
type filter text
Name
Select A
Details
Add Repository
Local...
Archive...
Cancel
OK
What is already installed?

Name: Eclipse Helios Plugins

Location: http://download.eclipse.org/releases/helios

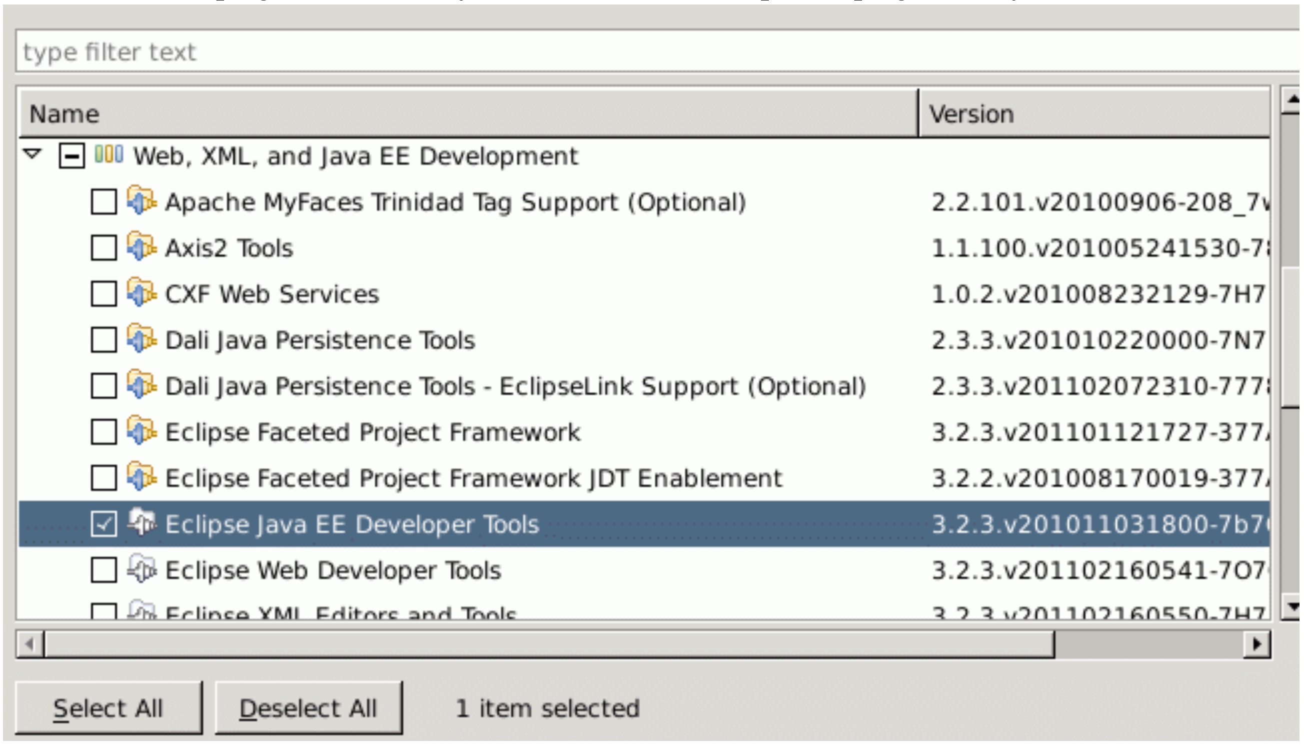
☑ Show only the latest versions of available software ☐ Hide items that are already installed

☑ Group items by category

☑ Contact all update sites during install to find required software

You can install plug-ins for Eclipse to add extra functionality that is not included in the core program.

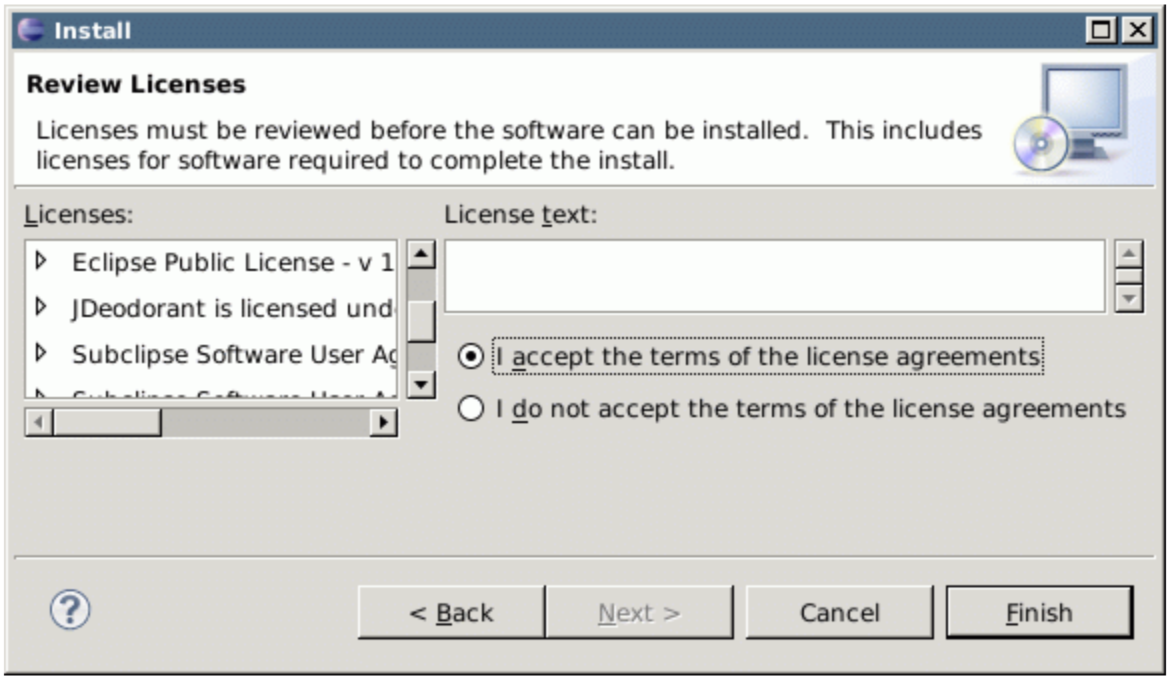
1. In the "Help" menu, select "Install new software..."
2. Select a plug-in repository from the "Work with:" menu. If the repository you need is not listed, follow the four steps below. Otherwise, you can skip those.
3. Click "Add..."
4. In the "Name" field, type the name of what you are trying to add. What you type does not matter- this will only help you recognize the plug-in source in the future.
5. In the "Location" field, type in the address of the plug-in repository you want to add.
   - For Eclipse Helios, the main repository is http://download.eclipse.org/releases/helios .
   - For Eclipse Galileo, the main repository http://download.eclipse.org/webtools/updates .
   - You can install Perl-related plugins with http://e-p-i-c.sf.net/updates .
   - Other plug-in repositories can be found through search engines.
6. Click "OK". You may need to wait a minute or two for the list of plug-ins to download.
7. When the list of plug-ins is shown, you can search for the specific plug-in that you want to install.



type filter text

Name	Version
Web, XML, and Java EE Development	
☐ Apache MyFaces Trinidad Tag Support (Optional)	2.2.101.v20100906-208_7v
☐ Axis2 Tools	1.1.100.v201005241530-7
☐ CXF Web Services	1.0.2.v201008232129-7H7
☐ Dali Java Persistence Tools	2.3.3.v201010220000-7N7
☐ Dali Java Persistence Tools - EclipseLink Support (Optional)	2.3.3.v201102072310-777
☐ Eclipse Faceted Project Framework	3.2.3.v201101121727-377
☐ Eclipse Faceted Project Framework JDT Enablement	3.2.2.v201008170019-377
✓ Eclipse Java EE Developer Tools	3.2.3.v201011031800-7b7
☐ Eclipse Web Developer Tools	3.2.3.v201102160541-7O7
☐ Eclipse XML Editors and Tools	3.2.3.v201102160550-7H7

8. In the example above, the Java EE Developer Tools plug-in will be installed.
9. Click "Next". If any of the plug-ins you are trying to install were already previously installed, you will be notified in the next window.
10. Click "Next" again, then (read and) accept the plug-in licenses, and click "Finish".


Install
Review Licenses
Licenses must be reviewed before the software can be installed. This includes licenses for software required to complete the install.
Licenses:
Eclipse Public License - v 1
JDeodorant is licensed und
Subclipse Software User Ag
License text:
I accept the terms of the license agreements
I do not accept the terms of the license agreements
< Back
Next >
Cancel
Finish

11. You may be notified by a security warning if the plug-in you are installing is unsigned. Generally that means the author is not verified, but in most cases, it is safe to continue.
12. Restart Eclipse

Installed Plugins

For a list of system-installed plugins on the ugrad installation of Eclipse, open up Eclipse on a ugrad computer, go to Help >> Install New Software ... and click on What is already installed?