Java Development with Eclipse
Lab 1 Getting Familiar with Eclipse

1. Creating the Project, package and java classes

Create a Project, **File**>**New**>**Project**,

1. Select Project type ‘Java Project’. Select **Next**.
2. Project name is ‘Lab’. Select **Finish**.

Create a Package, **File**>**New**>**Package**,

1. Enter ‘Lab’ in ‘Source Folder’ text window.
2. Enter ‘eclipseLab’ in ‘name’ text window. Select **Finish**.

Create a Class, **File**>**New**>**Class**,

1. Enter ‘Lab’ in ‘Source Folder’ text window.
2. Enter ‘eclipseLab’ in ‘Package’ window.
3. Enter ‘CountPrinter’ in ‘name’ window.
4. Unselect all of the ‘tick boxes’ in the bottom 3 boxes.
5. Select **Finish**

Enter the following code in the generated ‘CountPrinter’ class template:

```
package eclipseLab;

public class CountPrinter {
      public CountPrinter(int num)
      {
            System.out.println(“Count = “ num);
      }
}
```

Create another Class, **File**>**New**>**Class**,

1. Enter ‘Lab’ in ‘Source Folder’ text window.
2. Enter ‘eclipseLab’ in ‘Package’ window.
3. Enter ‘RunIt’ in ‘name’ window.
4. Tick the ‘public static void main(String[] args)’ ‘tick box’
5. Select in **Finish**

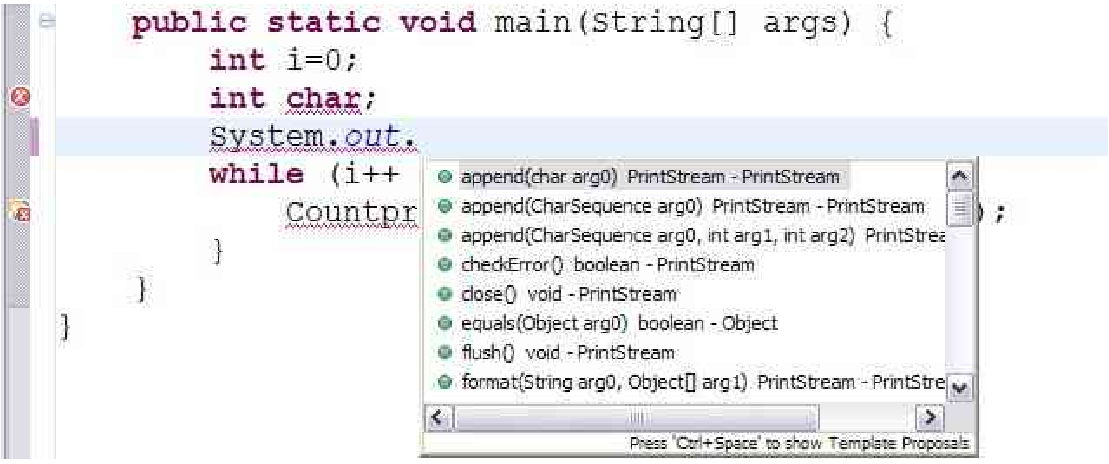
Enter the following code in the generated ‘RunIt’ class template:

```
package eclipseLab;

public class RunIt {

      public static void main(String[] args) {
            int i=0;
            int char;
            System.out.println("Lab1");
            while (i++ < 10) {
                  Countprinter cp = new CountPrinter(i);
```

```
                }
            }
        }
```

2. Code Assist

When entering ‘System.out.println’ notice how after entering ‘System’ and ‘System.out.’ Eclipse displays a list of the objects and methods available in the System and System.out Classes respectively. System is a java class which enables some properties and behaviour of the Java Virtual Machine to be configured and retrieved programmatically. Out is a PrintStream object which is an attribute of System and is the standard output stream.



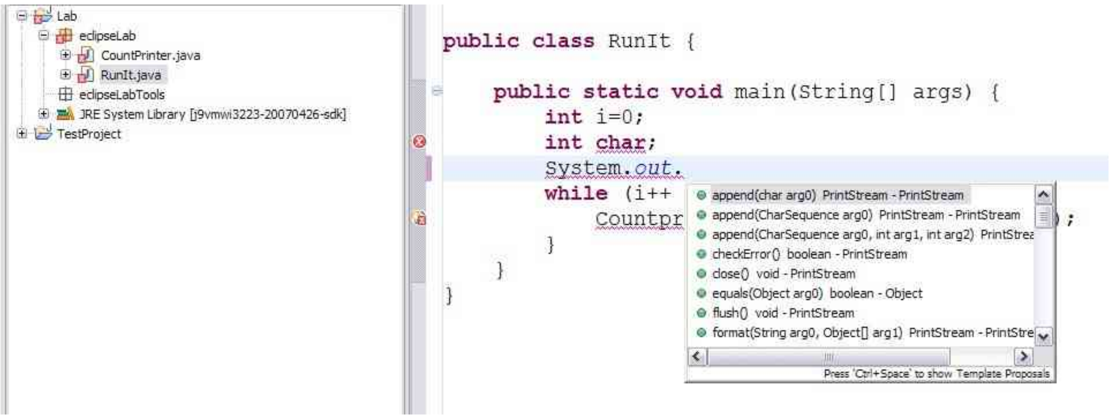
```
    public static void main(String[] args) {
        int i=0;
        int char;
        System.out.
        while (i++
            Countpr                                    );
        }
    }
}
```

append(char arg0) PrintStream - PrintStream
append(CharSequence arg0) PrintStream - PrintStream
append(CharSequence arg0, int arg1, int arg2) PrintStrea
checkError() boolean - PrintStream
close() void - PrintStream
equals(Object arg0) boolean - Object
flush() void - PrintStream
format(String arg0, Object[] arg1) PrintStream - PrintStre
Press 'Ctrl+Space' to show Template Proposals

3. Correcting the code errors

Code errors have been ‘placed’ in the code to demonstrate:

1. Icons associated with errors and warnings. Hint, hold the mouse cursor over the error / warning icon for more information.
2. Places where errors and warnings are identified
3. Demonstrate the Quick-Fix function
4. Compilation is done automatically (on file save)



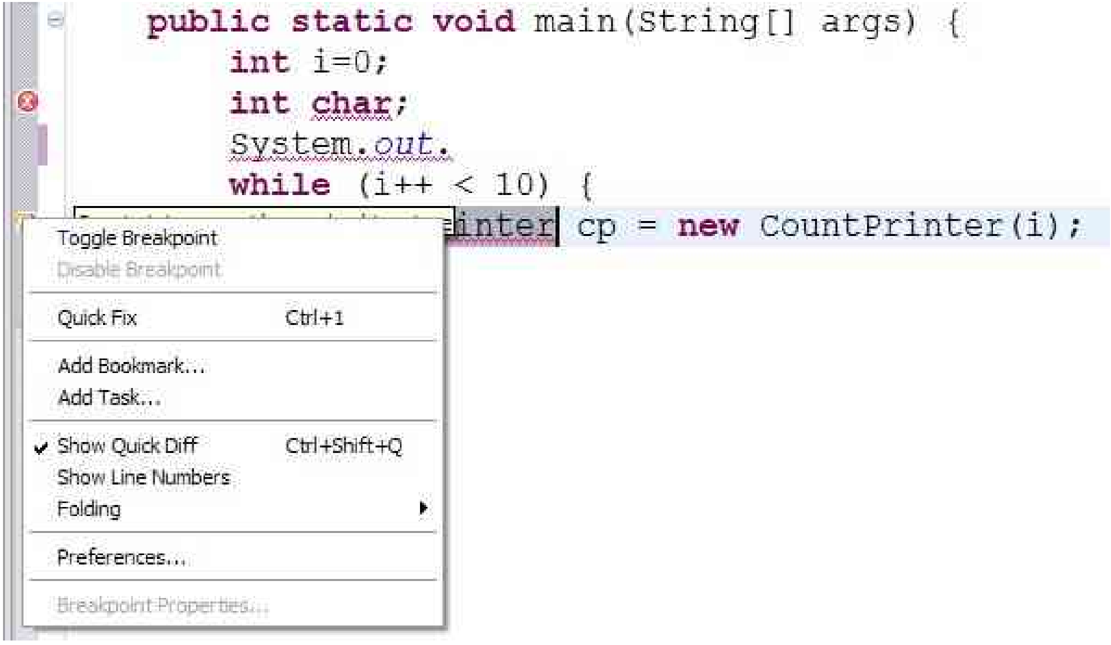
```
public class RunIt {

    public static void main(String[] args) {
        int i=0;
        int char;
        System.out.
        while (i++
            Countpr                                    );
        }
    }
}
```

Lab
eclipseLab
CountPrinter.java
RunIt.java
eclipseLabTools
JRE System Library [j9vmwi3223-20070426-sdk]
TestProject
append(char arg0) PrintStream - PrintStream
append(CharSequence arg0) PrintStream - PrintStream
append(CharSequence arg0, int arg1, int arg2) PrintStrea
checkError() boolean - PrintStream
close() void - PrintStream
equals(Object arg0) boolean - Object
flush() void - PrintStream
format(String arg0, Object[] arg1) PrintStream - PrintStre
Press 'Ctrl+Space' to show Template Proposals

The errors, and corresponding code changes, are:

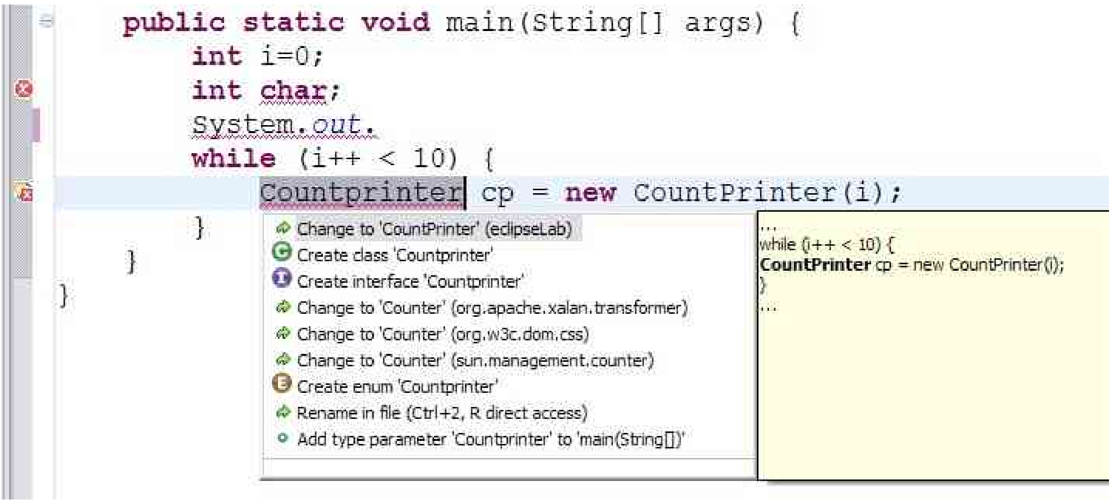
In RunIt.java

1. Remove ‘int char;’ char is a reserve word and can’t be used as an identifier.
2. Right-Click on the error icon on the Countprinter cp = new CountPrinter(i) line
   Select **Quick-Fix**



```
public static void main(String[] args) {
    int i=0;
    int char;
    System.out.
    while (i++ < 10) {
        inter cp = new CountPrinter(i);
```

Select “`Change to ‘CountPrinter’ (eclipseLab)`”



```
public static void main(String[] args) {
    int i=0;
    int char;
    System.out.
    while (i++ < 10) {
        Countprinter cp = new CountPrinter(i);
    }
  }
}
```

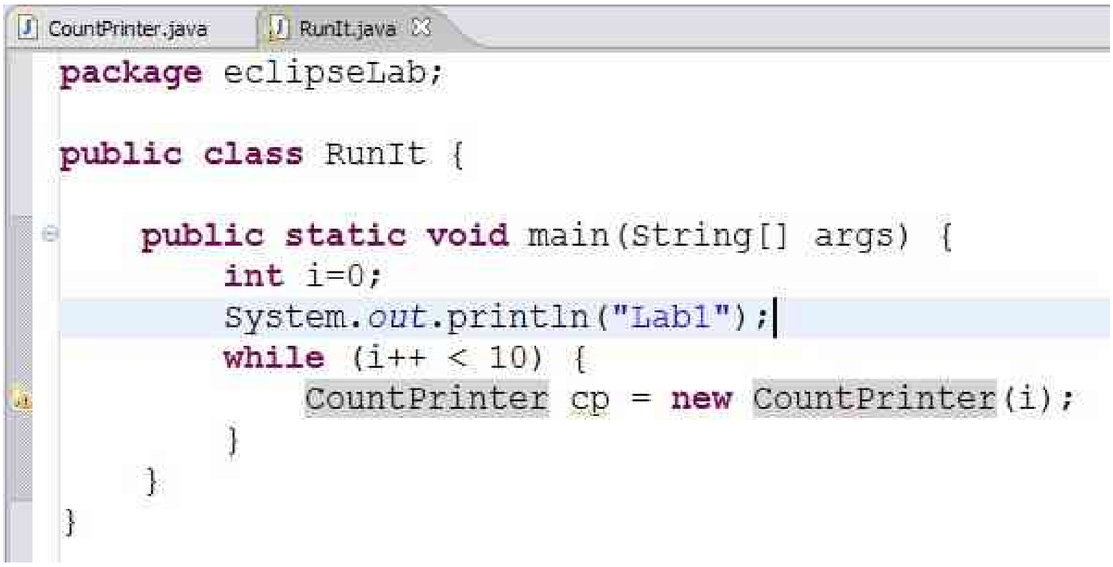
Change to 'CountPrinter' (eclipseLab)
Create class 'Countprinter'
Create interface 'Countprinter'
Change to 'Counter' (org.apache.xalan.transformer)
Change to 'Counter' (org.w3c.dom.css)
Change to 'Counter' (sun.management.counter)
Create enum 'Countprinter'
Rename in file (Ctrl+2, R direct access)
Add type parameter 'Countprinter' to 'main(String[])'

```
...
while (i++ < 10) {
CountPrinter cp = new CountPrinter();
}
...
```

In CountPrinter.java

1. Change the print line to the following:
   `System.out.println("Count = " + num);`

4. For this exercise don’t worry about the warning



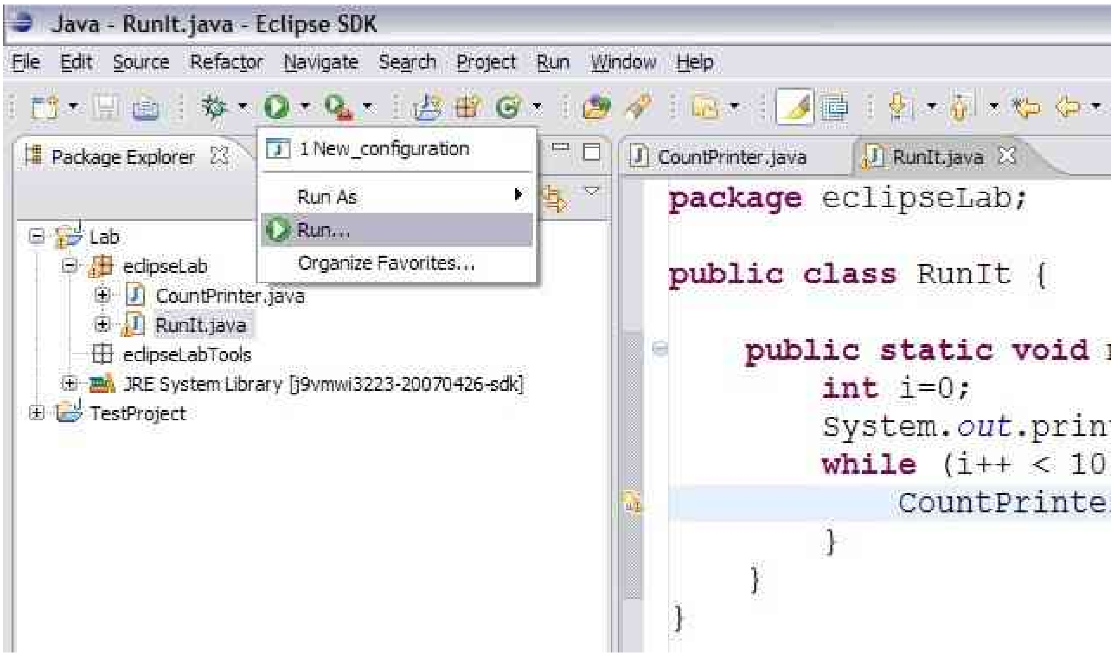
```
package eclipseLab;

public class RunIt {

    public static void main(String[] args) {
        int i=0;
        System.out.println("Lab1");
        while (i++ < 10) {
            CountPrinter cp = new CountPrinter(i);
        }
    }
}
```

`cp` is written to but never read.

5. Running the code

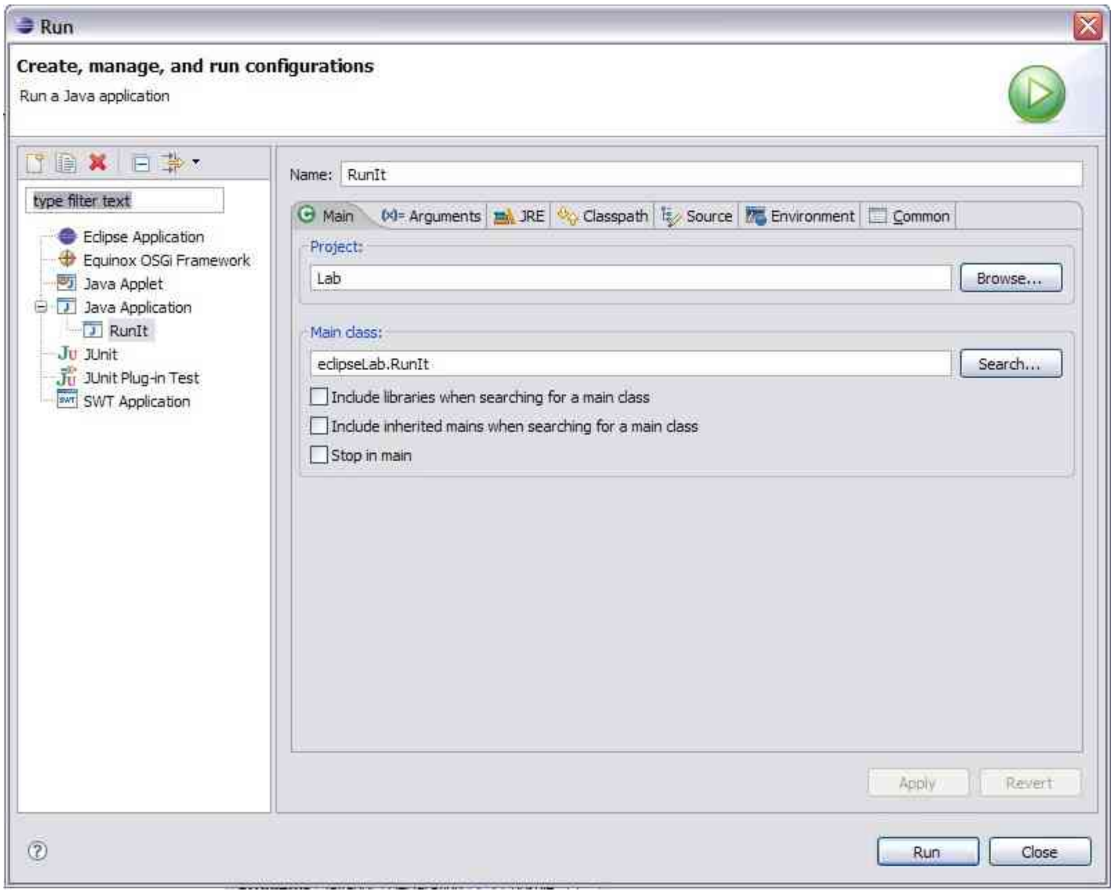
To run the application eclipseLab.RunIt( )

1. ‘Click’ the drop down arrow next to the run button. Select **Run…**



```
package eclipseLab;

public class RunIt {

    public static void
        int i=0;
        System.out.prin
        while (i++ < 10
            CountPrinte
        }
    }
}
```

This will bring up the following panel:



Create, manage, and run configurations

Run a Java application

Name: RunIt

Project: Lab

Main class: eclipseLab.RunIt

☐ Include libraries when searching for a main class

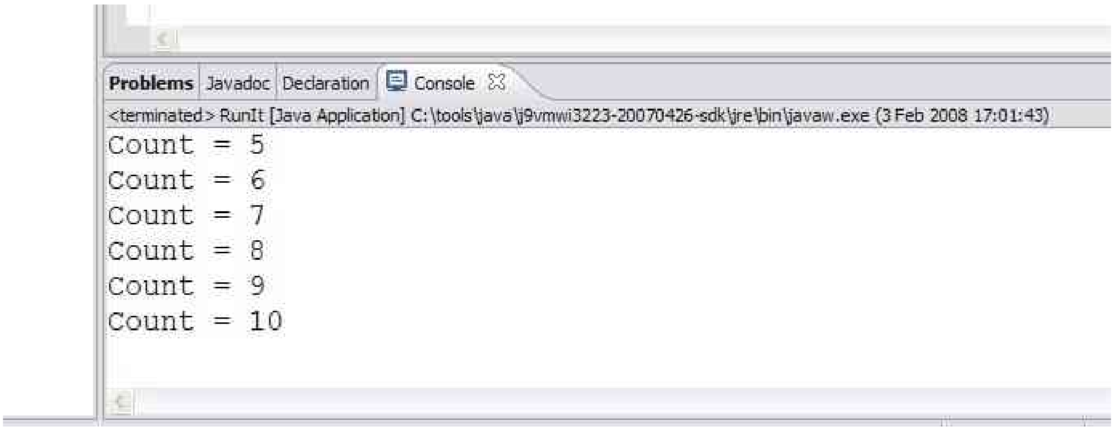
☐ Include inherited mains when searching for a main class

☐ Stop in main

Confirm that Main class is eclipseLab.RunIt

Click on the **Run** button…

Click on the ‘Console’ tab at the bottom of your eclipse workspace to see the output from your code:



```
<terminated> RunIt [Java Application] C:\tools\java\j9vmwi3223-20070426-sdk\jre\bin\javaw.exe (3 Feb 2008 17:01:43)
Count = 5
Count = 6
Count = 7
Count = 8
Count = 9
Count = 10
```

If you have time…

Try these other cool features if you have some time left or do them in your own time. Each of the following sections are independent so do them in any order you wish.

6. Refactoring

The goal of refactoring is to allow you to make system-wide code changes without affecting the semantic behaviour of the system. In this section, we will create a new package, and move the CountPrinter Class into it, and allow Eclipse to update all references to it.

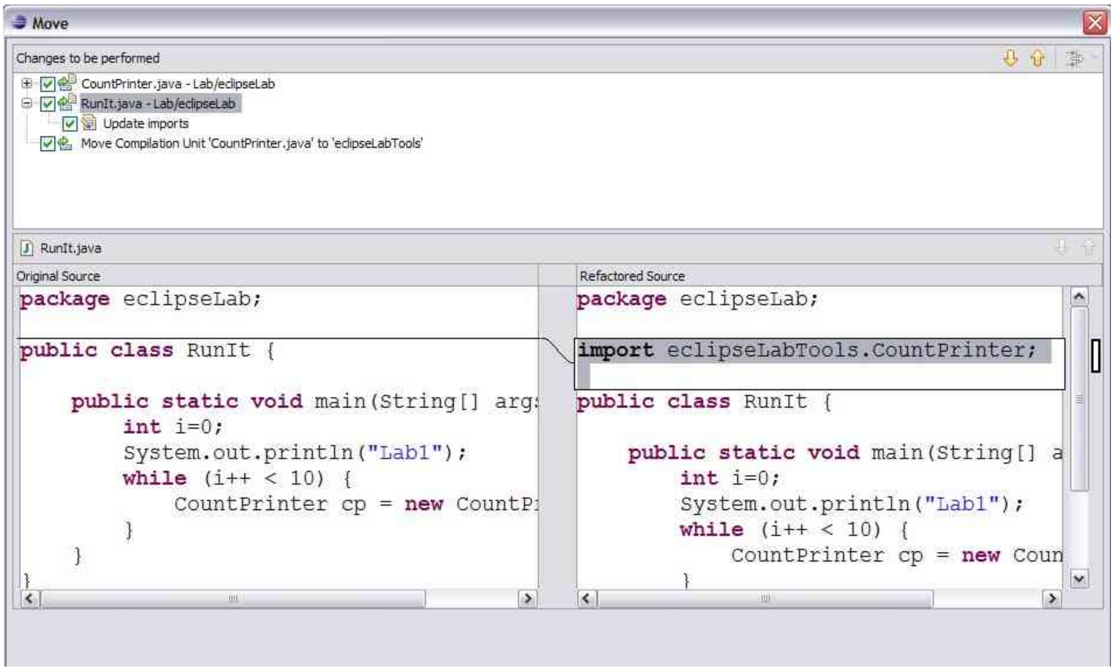
Create a Package, **File**>**New**>**Package**,

1. Enter ‘Lab’ in ‘Source Folder’ text window.
2. Enter ‘eclipseLabTools’ in ‘name’ text window. Select **Finish**.

Refactor the Code

1. Right-Click on the CountPrinter.java Class in the PackageExplorer.
2. Select **Refactor>Move**.
3. In the Move pop-up select eclipseLabTools and click **preview** >.

   Note the changes to RunIt.java:



☑ CountPrinter.java - Lab/eclipseLab
☑ RunIt.java - Lab/eclipseLab
  ☑ Update imports
☑ Move Compilation Unit 'CountPrinter.java' to 'eclipseLabTools'

Original Source

```
package eclipseLab;

public class RunIt {

    public static void main(String[] arg
        int i=0;
        System.out.println("Lab1");
        while (i++ < 10) {
            CountPrinter cp = new CountP
        }
    }
}
```

Refactored Source

```
package eclipseLab;

import eclipseLabTools.CountPrinter;

public class RunIt {

    public static void main(String[] a
        int i=0;
        System.out.println("Lab1");
        while (i++ < 10) {
            CountPrinter cp = new Coun
        }
```

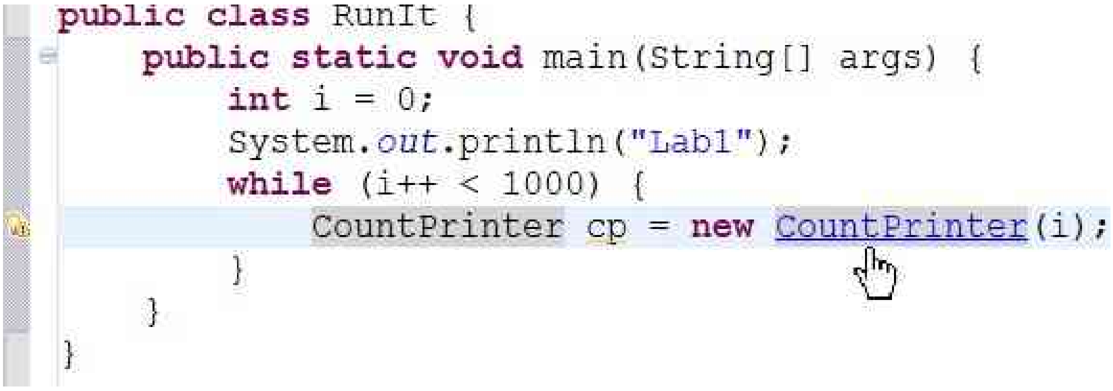
4. Review the changes that will be made then click **OK**.
5. See that a new import statement has been introduced into the RunIt Class and CountPrinter has been moved.

6. Try using refactoring to change the name of the CountPrinter Class.

7. Declarations and References

- Eclipse has tooling to allow users to find out where the definition of any method is. This is a declaration.

    1. To find a the CountPrinter method declaration, open the RunIt.java file in Eclipse.
    2. Hold ctrl down and, using the mouse left click on a CountPrinter call. See how it skips to the CountPrinter file.

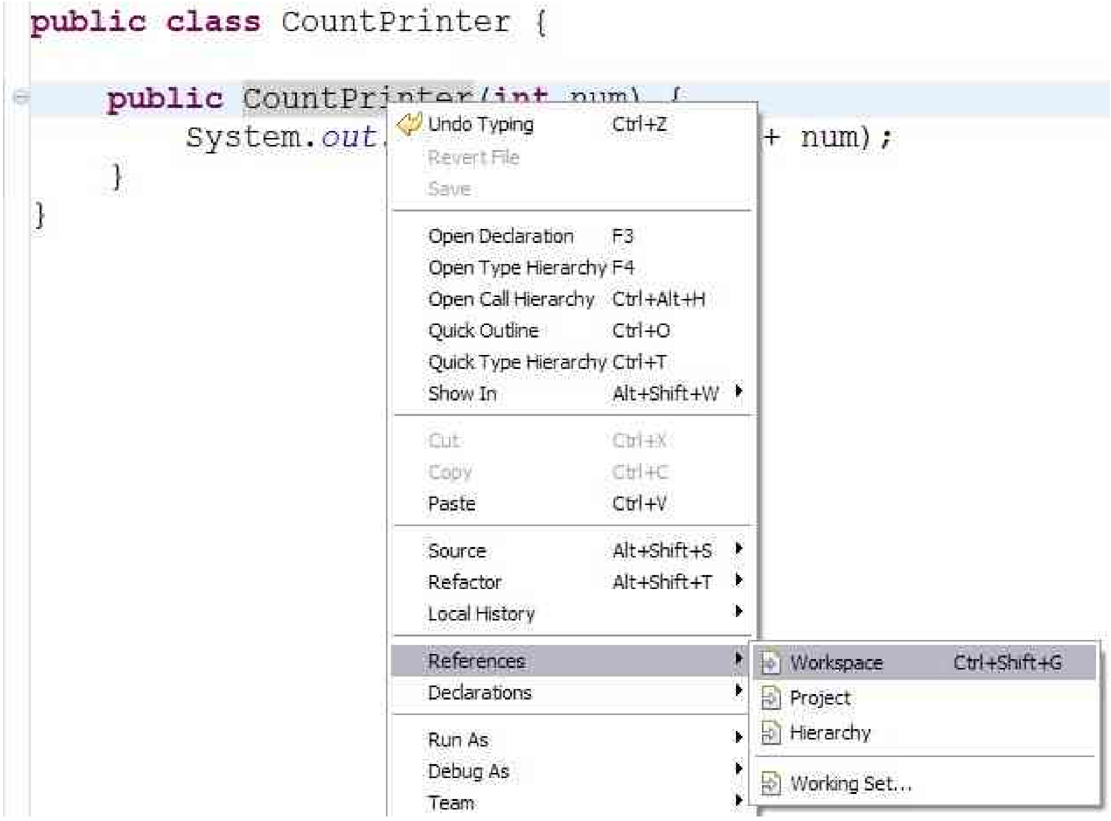


```
public class RunIt {
    public static void main(String[] args) {
        int i = 0;
        System.out.println("Lab1");
        while (i++ < 1000) {
            CountPrinter cp = new CountPrinter(i);
        }
    }
}
```

    3. Try looking for the println(…) definition. What source code file does it find?

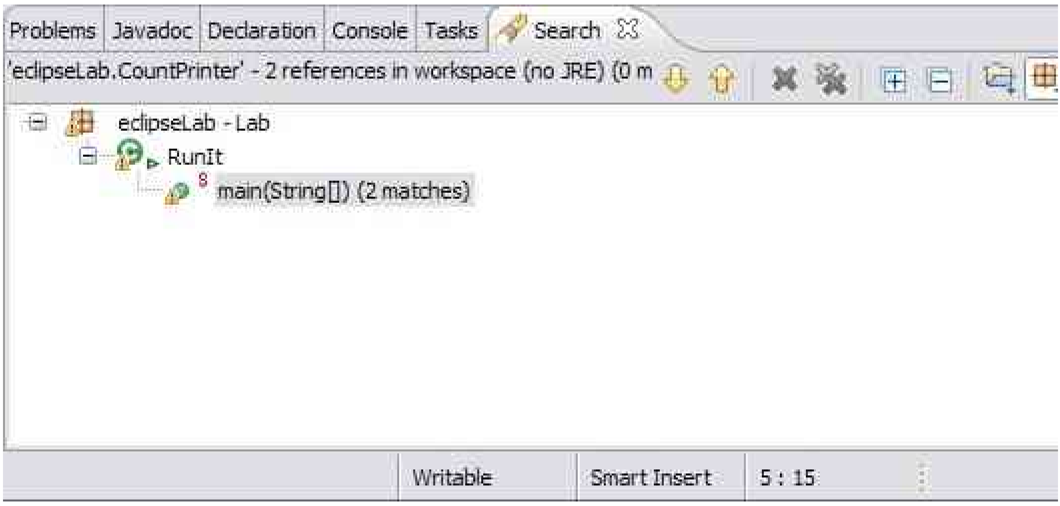
- Eclipse has tooling to allow users to list all calls made in a Project, Workspace or Hierarchy to any given method. These are references.

    1. To find all calls to any given method, open the CountPrinter.java file in Eclipse.
    2. Right-Click on the CountPrinter constructor method, select References>Workspace. See how a search panel opens listing the call to CountPrinter from RunIt.java.



```
public class CountPrinter {

    public CountPrinter(int num) {
        System.out.                    + num);
    }
}
```

3. Double-Click the entry on the search panel to go to any given occurrence



'eclipseLab.CountPrinter' - 2 references in workspace (no JRE) (0 m

eclipseLab - Lab
RunIt
main(String[]) (2 matches)

.

8. So what is this Javadoc thing all about?

Javadoc is a tool for generating documentation just like Sun’s API documentation **http://java.sun.com/j2se/javadoc/**

You can create your own documentation using the facilities provided in the Java language. The most important thing to note is the use of the special Javadoc comment commands, **/**** and ***/**.

So, as a reference, here is some useful syntax:

```
/**
 * Javadoc documentation goes between the opening and
 * closing tags and can spread over multiple lines.
 */
```

Tags include:

```
@author
@exception
@param
@return
@throws
```

A Javadoc comment could look something like:

```
/**
 * CountPrinter provides a method to print out a number
 * @author fentono
 */
public class CountPrinter {
…
```

1. Document the CountPrinter class

Add a Javadoc comment for the CountPrinter class and another just for the CountPrinter method in that class. Add several of the tags shown above and view the Javadoc in the Javadoc window.

2. Export the Javadoc and view in a browser

It's easy to take the Javadoc from your code and export it as html ready for a browser.

- Select File>Export>Javadoc.
- Ensure CountPrinter.java is selected
- Click Finish and accept any prompts

Several files should have been created in your workspace. Look for index.html and double-click. You are now in a standard browser environment (Eclipse has one built-in), select the CountPrinter class and view the Javadoc you created.