

CS2334 Lab2

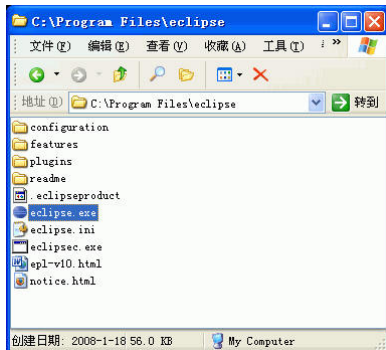
Eclipse And Debugging Survival Guide

Yu-Hsin Li and Mark Woehrer

Jan. 24, 08

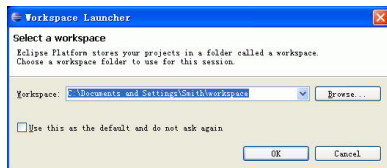
Basic Eclipse Tutorial

- ▶ Extract the file eclipse-SDK-3.6-win32.zip to C:\Program Files.
- ▶ Go to C:\Program Files\eclipse.
- ▶ Double click on eclipse.exe.

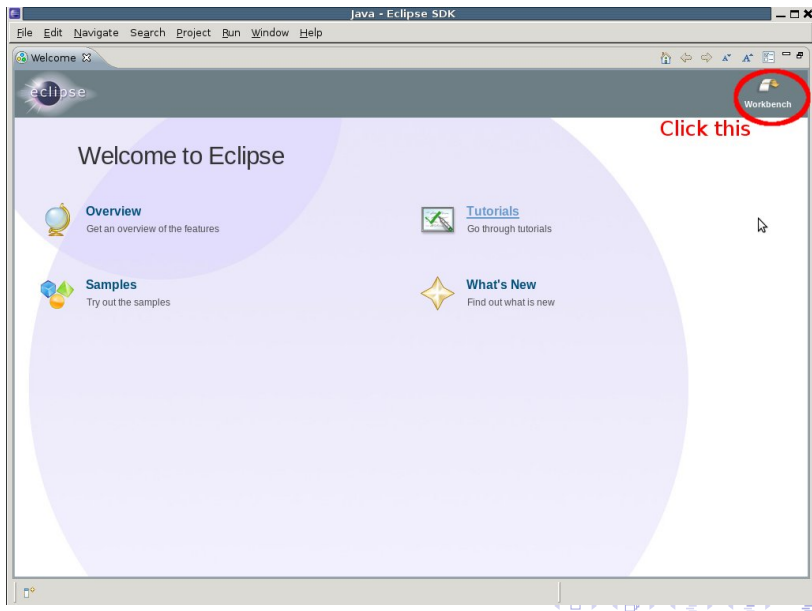


Choose a Workspace

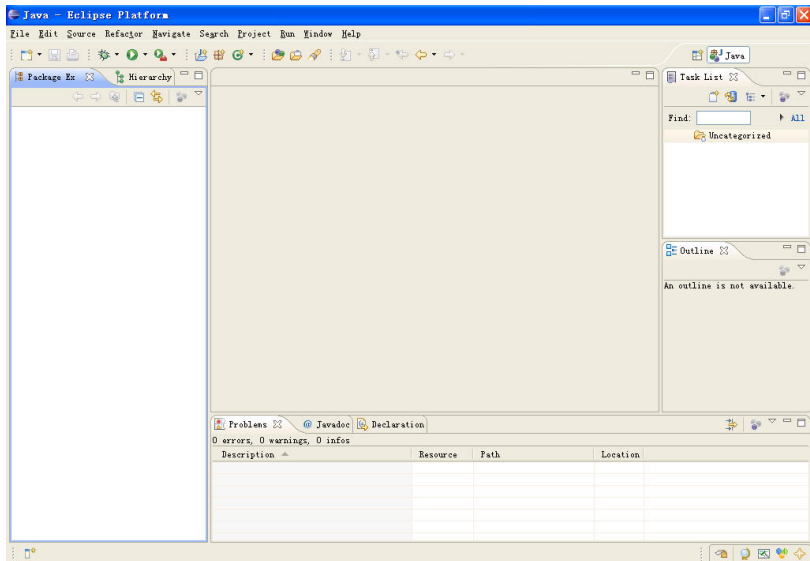
- ▶ A workspace is a folder/directory in your hard drive.
- ▶ Each workspace houses a collection of projects.
- ▶ Click OK for the default one.



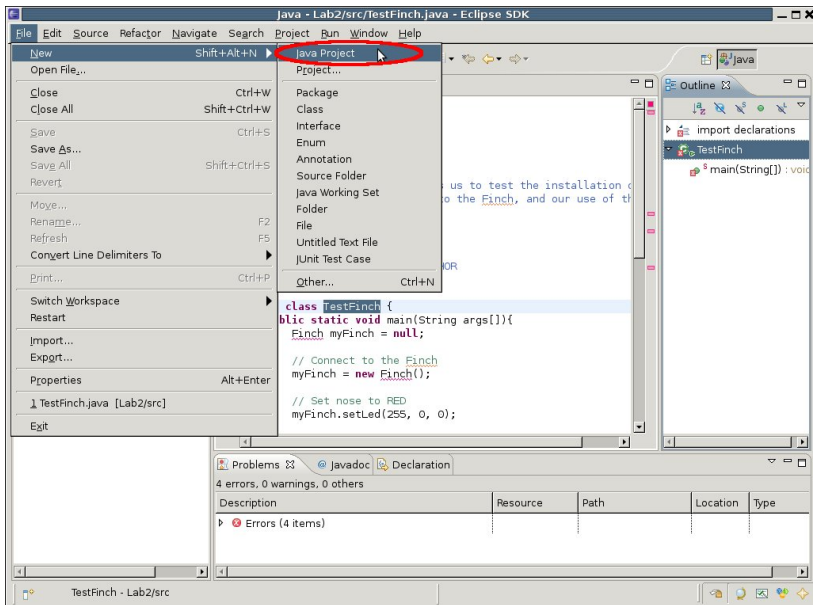
The Welcome Screen



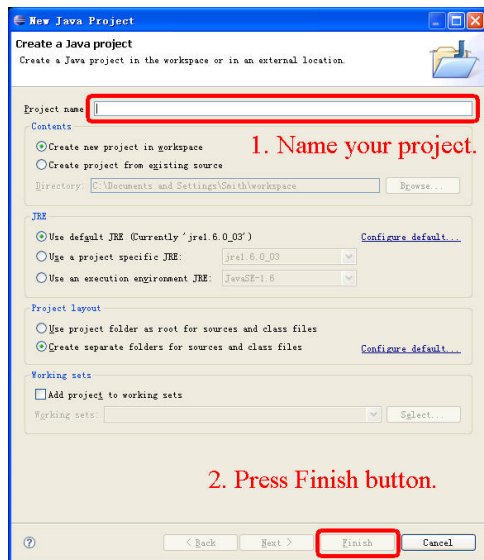
The Main Screen



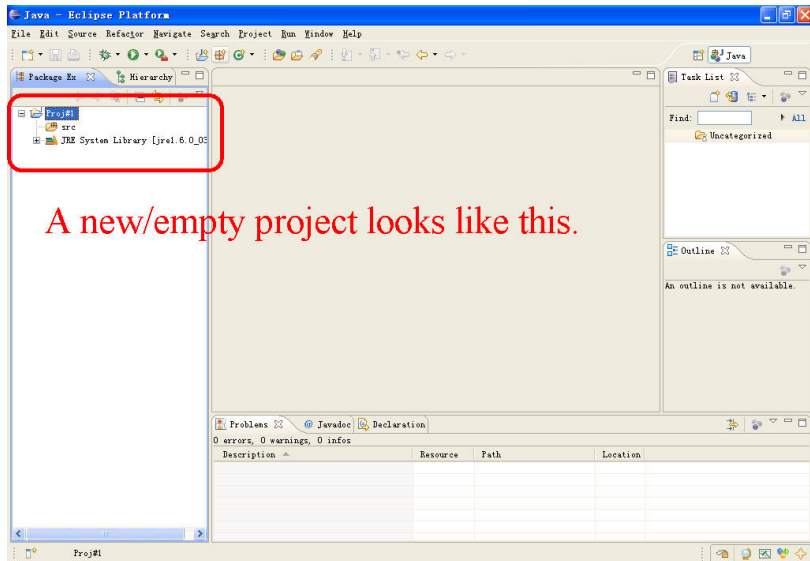
Create a New Project



Create a New Project (Cont'd)



Create a New Project (Cont'd)



Add a Class to a Project

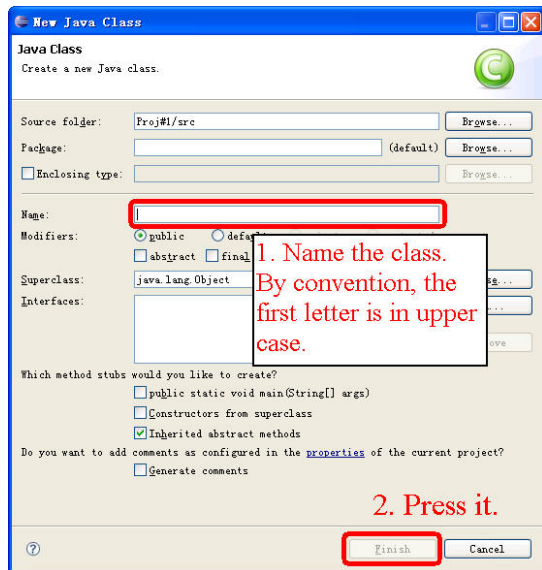
The screenshot shows the Eclipse IDE interface. In the Package Explorer on the left, a project named 'Proj#1' is selected and highlighted with a red box. A red arrow points from the text '1. Click on the project in which you want to add a class.' to this box. In the main editor area, the 'Add Class' button (represented by a green 'G' icon) is highlighted with a red box. A red arrow points from the text '2. Press this.' to this button. The bottom of the IDE shows the 'Problems' view with 0 errors, 0 warnings, and 0 infos, and a table with columns for Description, Resource, Path, and Location.

1. Click on the project in which you want to add a class.

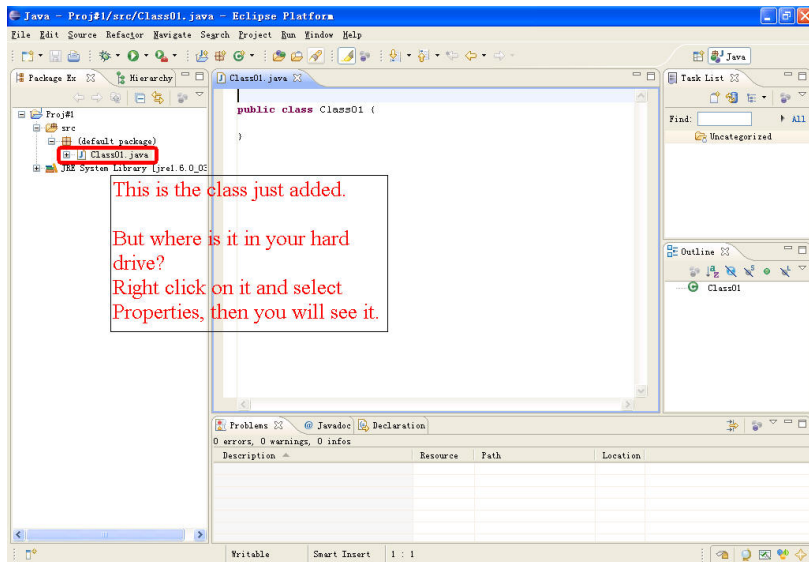
2. Press this.

Description	Resource	Path	Location

Add a Class to a Project (Cont'd)



Add a Class to a Project (Cont'd)



Java - Proj#1/src/Class01.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Package Ex Hierarchy

Proj#1

- src
- (default package)
- Class01.java**
- JRE System Library [jre1.6.0_05]

```
public class Class01 {  
  
}
```

Task List

Find: All

Uncategorized

Outline

Class01

Problems Javadoc Declaration

0 errors, 0 warnings, 0 infos

Description	Resource	Path	Location

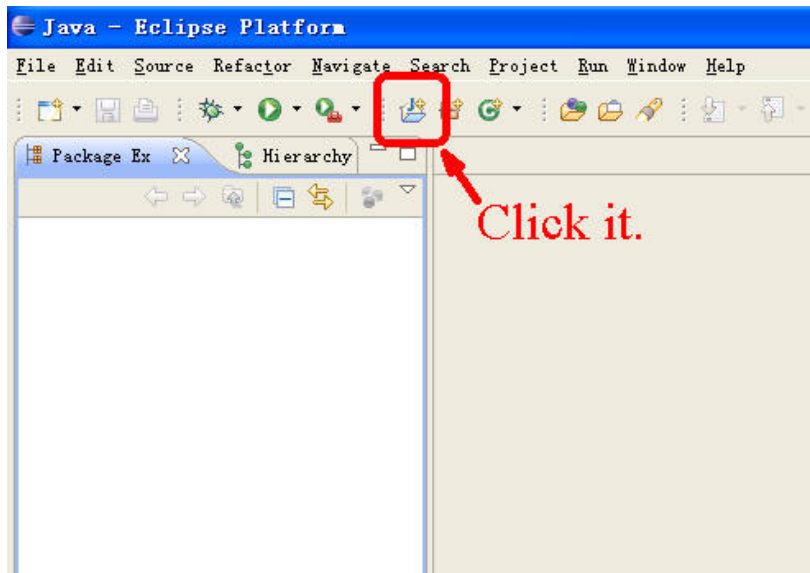
Writabile Smart Insert 1 : 1

This is the class just added.

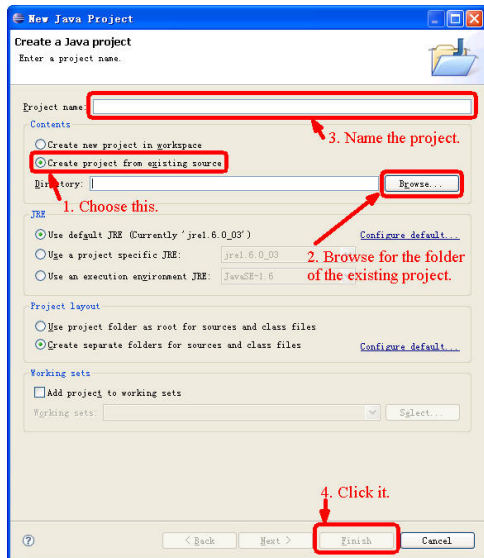
But where is it in your hard drive?

Right click on it and select Properties, then you will see it.

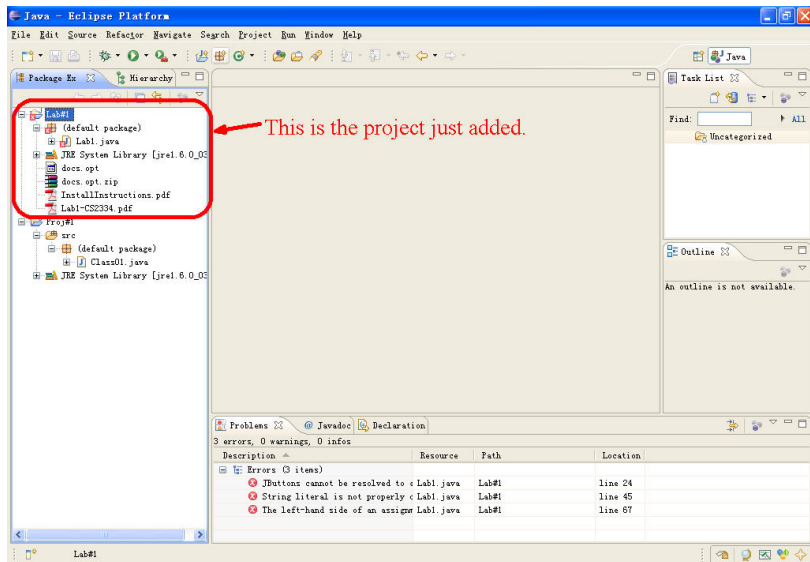
Add a Directory As a New Project



Add a Directory As a New Project (Cont'd)



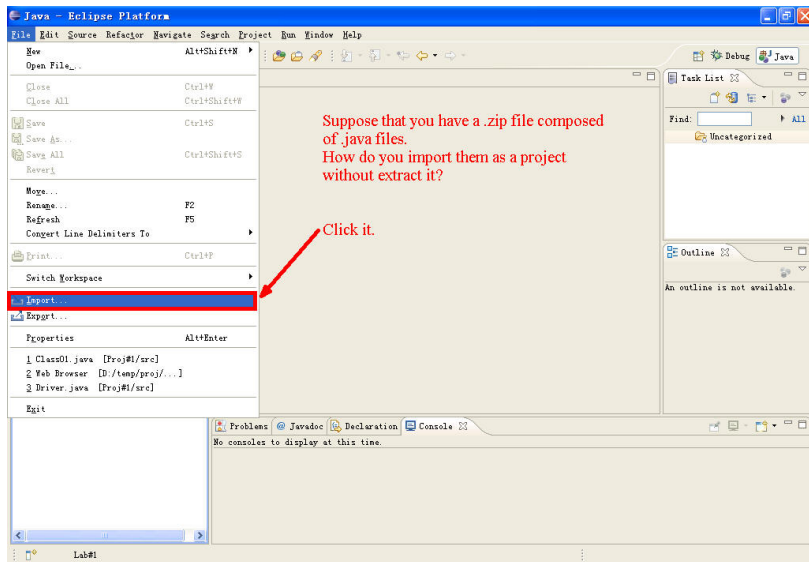
Add a Directory As a New Project (Cont'd)



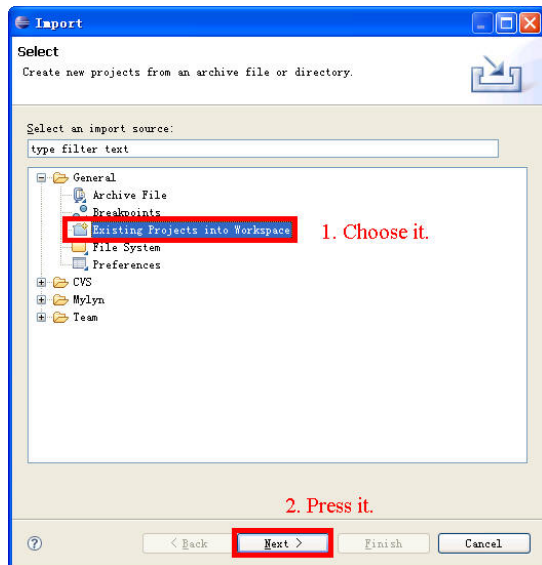
The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project named 'Lab#1' which is highlighted with a red box. A red arrow points to this box with the text 'This is the project just added.' The project structure includes a default package, Lab1.java, and a JRE System Library. Below the Package Explorer, the Problems view shows three errors:

Description	Resource	Path	Location
Errors (3 items)			
JBUTTONS cannot be resolved to a class in the current scope	Lab#1	Lab#1	line 24
String literal is not properly enclosed in quotes	Lab#1	Lab#1	line 45
The left-hand side of an assignment must be a variable, a field, or a property	Lab#1	Lab#1	line 67

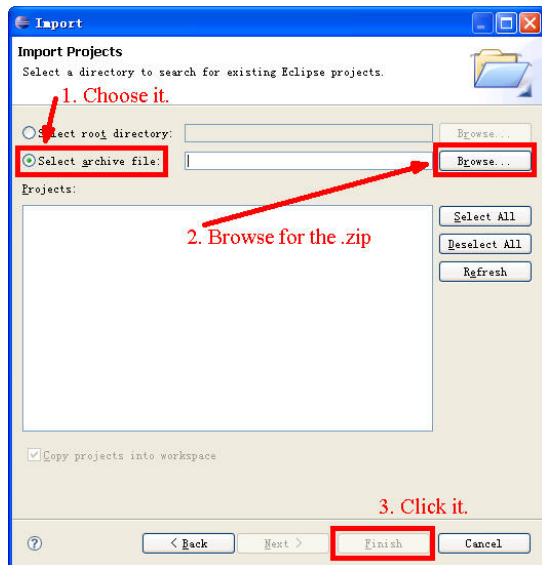
Add a .zip File As a New Project



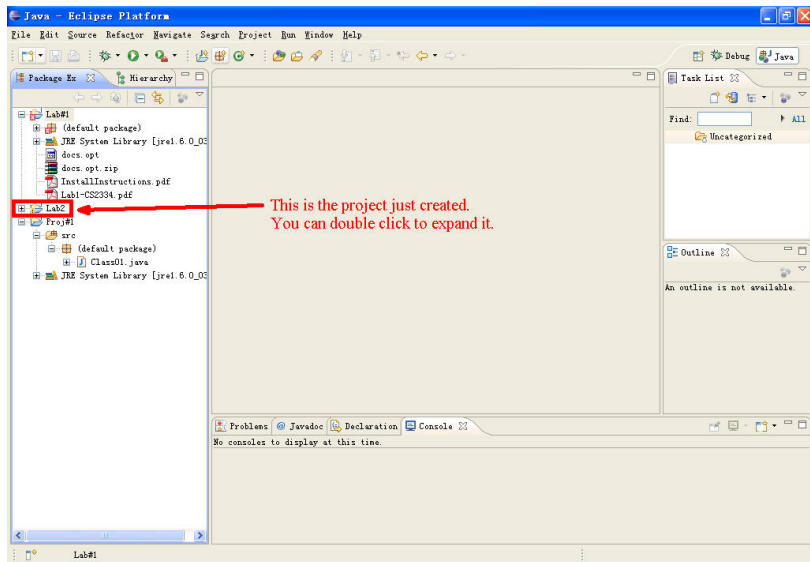
Add a .zip File As a New Project (Cont'd)



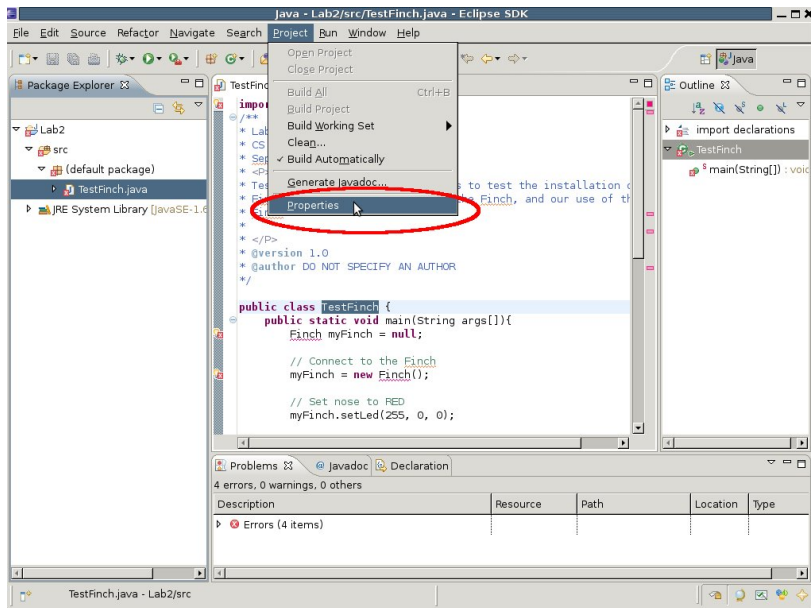
Add a .zip File As a New Project (Cont'd)



Add a .zip File As a New Project (Cont'd)



Adding an external .jar to a Project



The screenshot shows the Eclipse IDE interface. The 'Project' menu is open, and the 'Properties' option is highlighted with a red circle. The main editor displays the source code for 'TestFinch.java'. The Package Explorer on the left shows the project structure, and the Outline view on the right shows the class structure. The Problems view at the bottom indicates 4 errors.

```
import java.awt.*;
import java.awt.event.*;

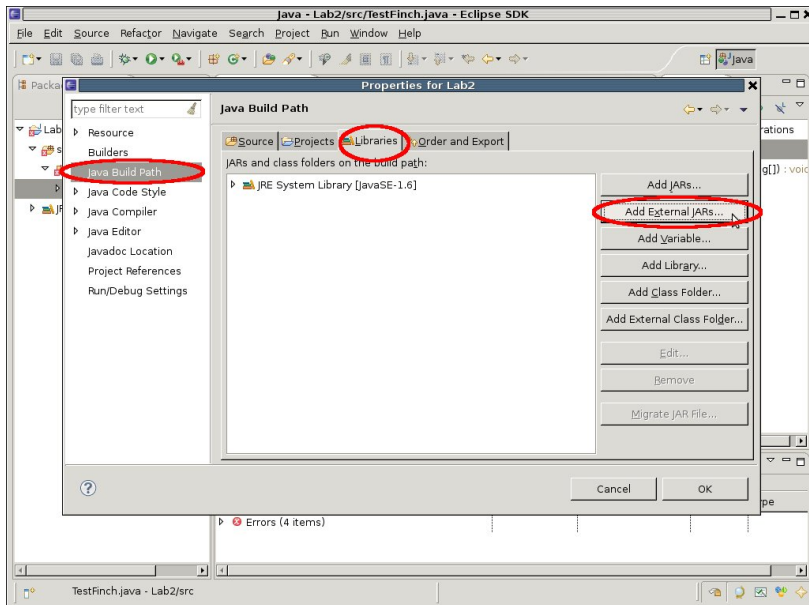
/**
 * TestFinch.java
 *
 * @version 1.0
 * @author DO NOT SPECIFY AN AUTHOR
 */

public class TestFinch {
    public static void main(String args[]){
        Finch myFinch = null;

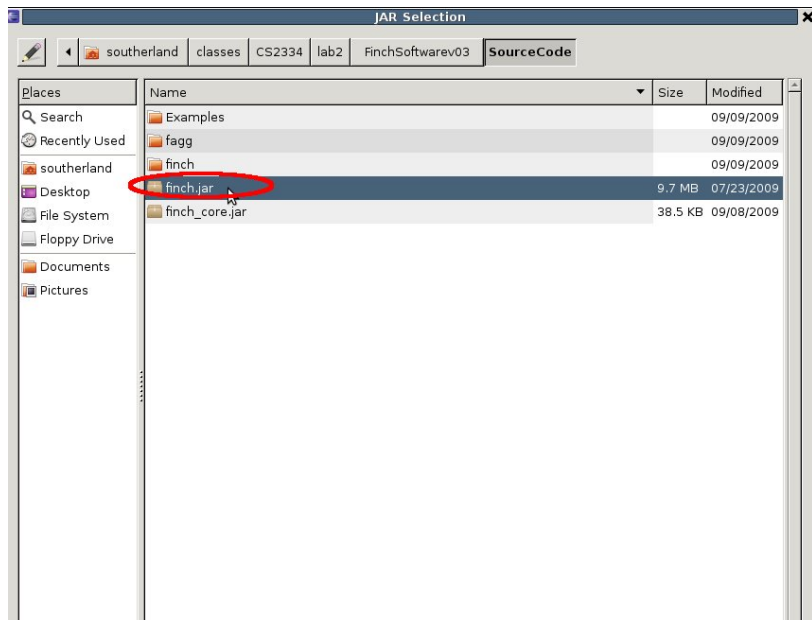
        // Connect to the Finch
        myFinch = new Finch();

        // Set nose to RED
        myFinch.setLed(255, 0, 0);
    }
}
```

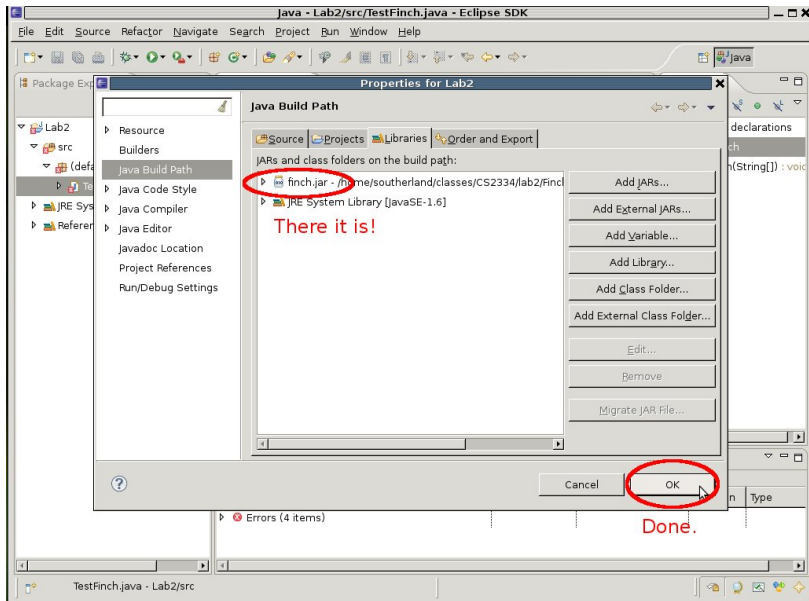
Adding an external .jar to a Project (Cont'd)



Adding an external .jar to a Project (Cont'd)



Adding an external .jar to a Project (Cont'd)



Locate Errors in a .java File

1. Double click on the filename which you want to see the content.

2. Then it will be shown here.

```
import javax.swing.*;

/**
 * Lab #1
 * CS 2334, Section *** INSERT YOUR LAB SECTION NUMBER HERE ***
 * *** THE CURRENT DATE GOES HERE ***
 * <P>
 * This class creates a sample program with a graphical user
 * interface.
 * </P>
 * @author *** YOUR NAME GOES HERE ***
 * @version 1.0
 */
public class Lab1 implements ActionListener
{
    /** The main window of the program. */
    private JFrame    windowFrame;
    /** A label that holds a message to be displayed in the wind
    private JLabel    textLabel;
    /** A button that will terminate the program and close the w
    private JButton    exitButton;

    /**

```

Description	Resource	Path	Location
Errors (3 items)			
JButton cannot be resolved to c.Lab1.java	Lab#1		line 24
String literal is not properly c.Lab1.java	Lab#1		line 45
The left-hand side of an assignw Lab1.java	Lab#1		line 67

Locate Errors in a .java File (cont'd)

The screenshot shows the Eclipse IDE with a Java file named `Lab1.java` open. The code contains several errors, which are highlighted with red circles and arrows. The `Problems` view at the bottom shows a list of these errors.

Annotations:

- Red circles and arrows point to error markers in the code editor.
- Text: "These are indices of errors. You can click on it." points to the error markers.
- Text: "Move mouse over here, Eclipse will pop up the error message." points to the Package Explorer.
- Text: "All errors are shown here." points to the Problems view.

Code Snippet:

```
/** The main window of the program. */
private JFrame windowFrame;
/** A label that holds a message to be displayed in the window. */
private JLabel textLabel;
/** A button that will terminate the program and close the window. */
JButtons cannot be resolved to a type JButtons;

/**
 * This is the constructor for the class Lab1. It initializes the main window of the program and sets up event handling for the program.
 * <P>
 * @param message The message to display to the user.
 */
public Lab1( String message )
{
    // Create a text label for the program.
    textLabel = new JLabel( message );

    /** Create a button for the program that will terminate the program and associate an action listener for the button.
     * and associate an action listener for the button.
     */
    exitButton = new JButton( "Click Here to Exit" );
}
```

Problems View:

Description	Resource	Path	Location
Errors (3 items)			
JButtons cannot be resolved to a type JButtons	Lab1.java	Lab1	line 24
String literal is not properly enclosed in quotes	Lab1.java	Lab1	line 45
The left-hand side of an assignment must be a variable	Lab1.java	Lab1	line 67

Locate Errors in a .java File (cont'd)

The screenshot shows the Eclipse IDE with a Java file named `Lab1.java` open. The code contains several errors, and a dialog box is open to help resolve one of them.

Code Snippet:

```
/** The main window of the program. */
private JFrame windowFrame;
/** A label that holds a message to be displayed in the window. */
private JLabel textLabel;
/** A button that will terminate the program and close the window. */
JButtons cannot be resolved to a type:
JButtons

/**
 * This is the main window of the program.
 * The message to be displayed in the window.
 * The button that will terminate the program and close the window.
 * <P>
 * @param args the command line arguments
 */
public Lab1(String message) {
    // Create the main window of the program.
    textLabel = new JLabel( message );

    /** Create a button for the program that will terminate the program
     * and associate an action listener for the button.
     */
    exitButton = new JButton( "Click Here to Exit" );
}
```

Dialog Box:

Creates the new class wizard to create the type.

Package: (default package)

```
public class JButtons {
```

Problems View:

Description	Resource	Path	Location
Errors (3 items)			
JButtons cannot be resolved to a type	Lab1.java	Lab1	line 24
String literal is not properly enclosed in quotes	Lab1.java	Lab1	line 45
The left-hand side of an assignment must be a variable or a field name	Lab1.java	Lab1	line 67

Annotations:

- A red circle highlights the `JButtons` error in the code.
- A red arrow points from the text "Click it. Eclipse will show possible solutions for you." to the red circle.
- A red box highlights the dialog box.

Compile a .java File

You don't need to do anything for compiling as long as this is checked.

You can also press "Ctrl + s" key to force Eclipse to compile the .java file.

```
the program. */
windowFrame;
message to be displayed in the window
label;
terminate the program and close the window
Button;
```

```
/* This is the constructor for the class Lab1. It initialize
 * the main window of the program and set ups event handling
 * the program.
 * <P>
 * @param message The message to display to the user.
 */
public Lab1( String message )
{
    // Create a text label for the program.
    textLabel = new JLabel( message );

    /* Create a button for the program that will terminate the
    * and associate an action listener for the button.
    */
    exitButton = new JButton( "Click Here to Exit" );
```

Description	Resource	Path	Location
✖ JButtons cannot be resolved to c:Lab1.java	Lab#1		line 24
✖ String literal is not properly c:Lab1.java	Lab#1		line 45
✖ The left-hand side of an assignm	Lab#1		line 67

Compile a .java File (cont'd)

The screenshot shows the Eclipse IDE with a Java file named `Lab1.java` open. The code defines a class `Lab1` with a constructor and several fields. The IDE's `Problems` view at the bottom shows three errors:

- `JButtons` cannot be resolved to a type in the current scope.
- String literal is not properly enclosed in quotes.
- The left-hand side of an assignment is not a variable or field.

A red arrow points from the text on the left to the `Problems` view.

```
/** The main window of the program. */
private JFrame windowFrame;
/** A label that holds a message to be displayed in the window. */
private JLabel textLabel;
/** A button that will terminate the program and close the window. */
private JButton exitButton;

/**
 * This is the constructor for the class Lab1. It initializes
 * the main window of the program and sets up event handling
 * for the program.
 * @param message The message to display to the user.
 */
public Lab1( String message )
{
    // Create a text label for the program.
    textLabel = new JLabel( message );

    // Create a button for the program that will terminate the
    // program and associate an action listener for the button.
    exitButton = new JButton( "Click Here to Exit" );
}
```

Eclipse is always compiling your .java file. So, if there is anything wrong, the error message will be shown here automatically.

How to Run a Project

1. Right click on the .java file containing the main() method.

```
import javax.swing.*;

/**
 * Lab #1
 */

/** INSERT YOUR LAB SECTION NUMBER HERE **
 * HERE GOES HERE **
 */

public class Lab1 extends JFrame implements ActionListener {

    // of the program. */
    private JFrame windowFrame;
    // adds a message to be displayed in the window
    private JLabel textLabel;
    // will terminate the program and close the window
    private JButton exitButton;

    // Declaration
}
```

2. Choose this.

	Location
Run As Java Application	line 85
Open Run Dialog...	

Run a Project with Command Line Arguments

1. Right click on the java file containing the main() method.

2. Click this.

```
INSERT YOUR LAB SECTION NUMBER HERE ***
GOES HERE ***

sample program with a graphical user

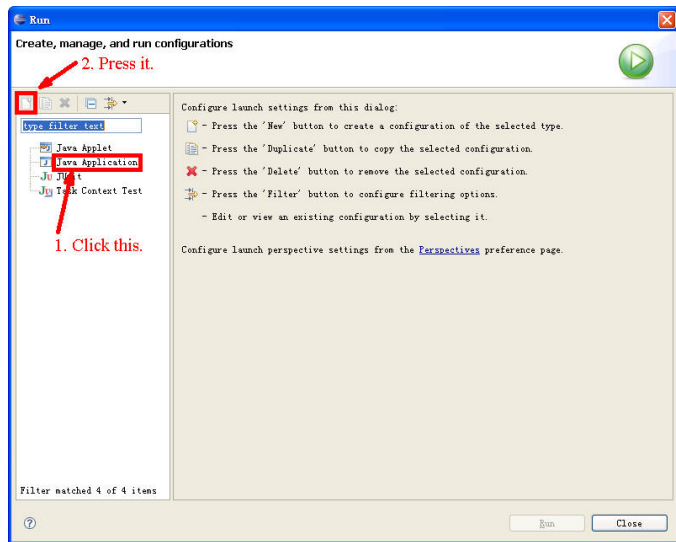
E GOES HERE ***

ments ActionListener

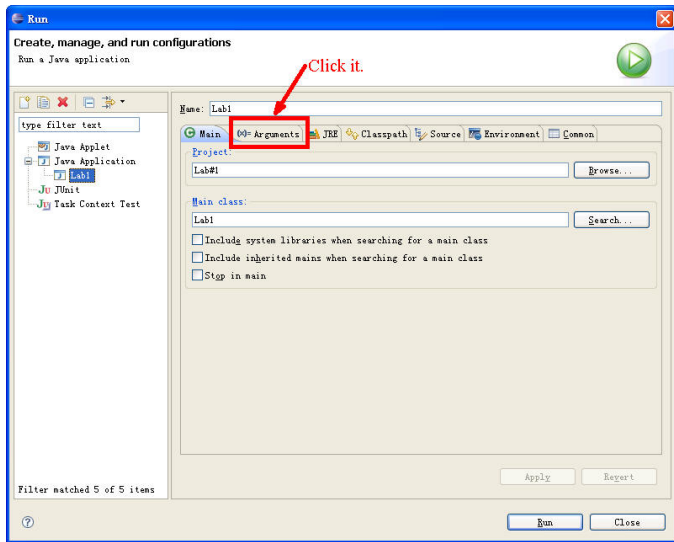
of the program. */
indowFrame;
ds a message to be displayed in the wind
extLabel;
ll terminate the program and close the w
xitButton;
```

Resource	Path	Location
1 Java Application	Alt+Shift+X, J	line 85

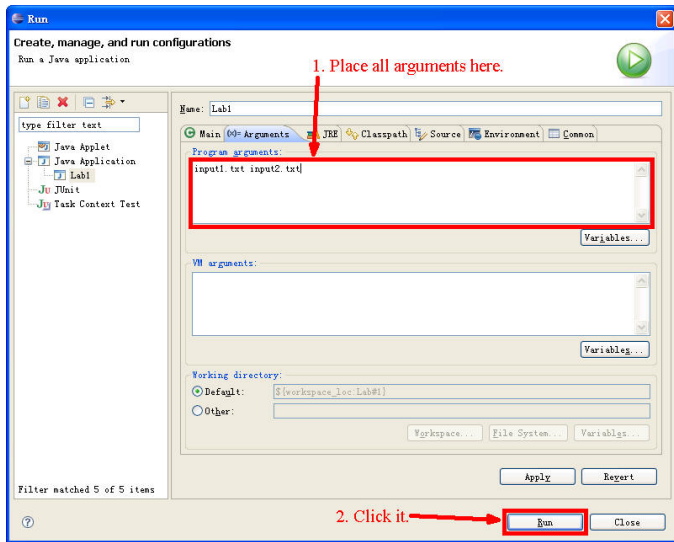
Run a Project with Command Line Arguments (cont'd)



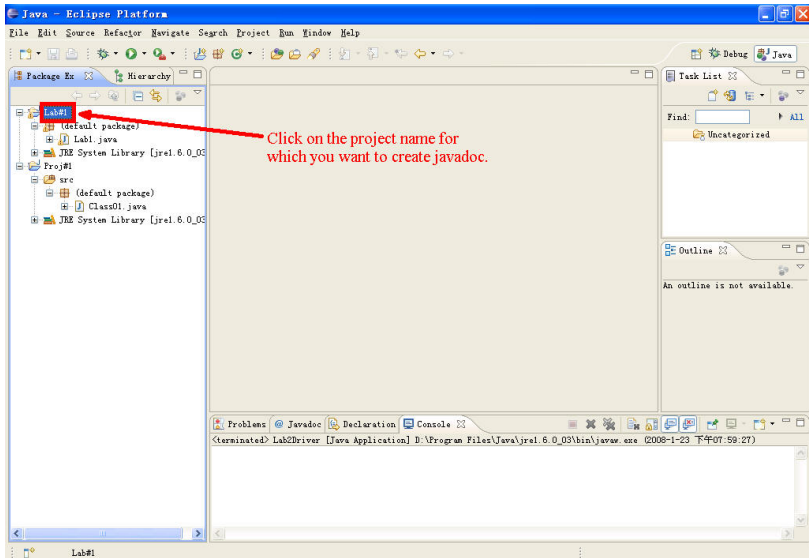
Run a Project with Command Line Arguments (cont'd)



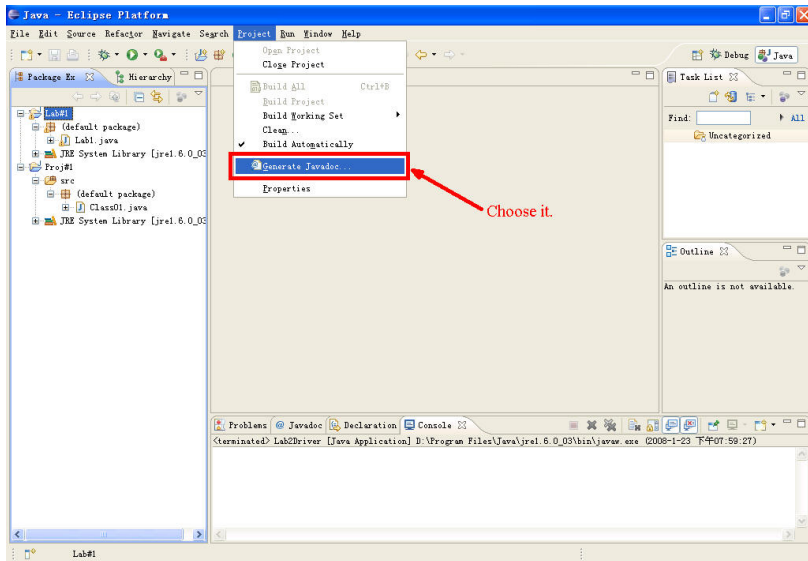
Run a Project with Command Line Arguments (cont'd)



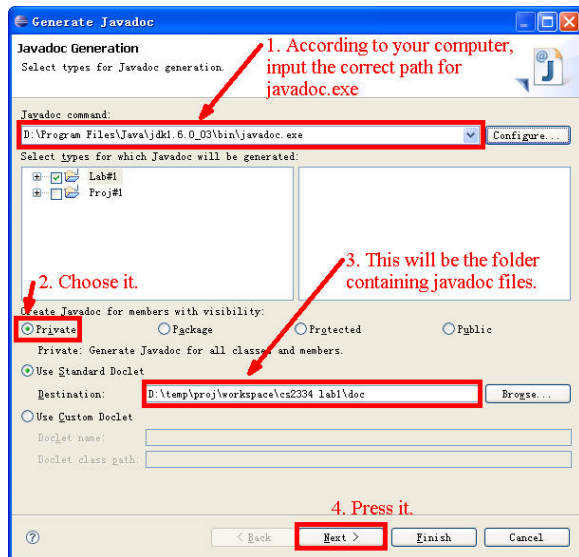
Create Javadoc



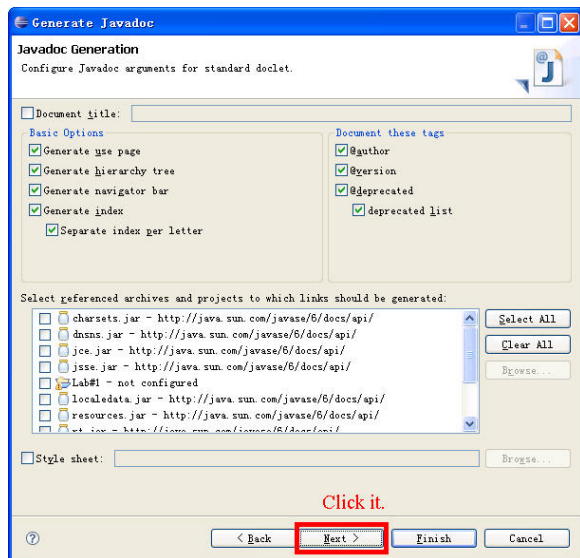
Create Javadoc (cont'd)



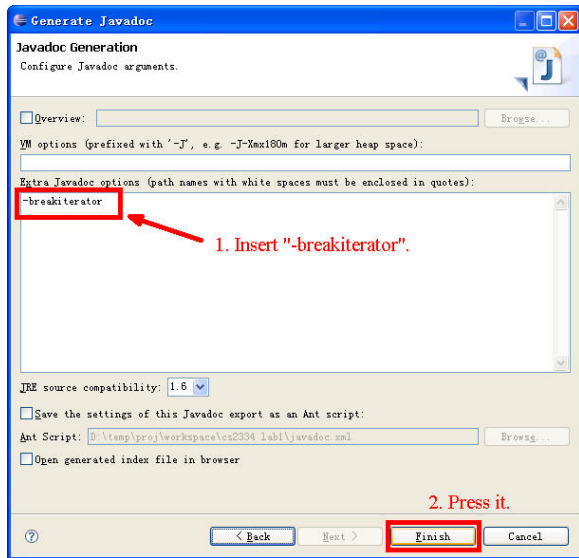
Create Javadoc (cont'd)



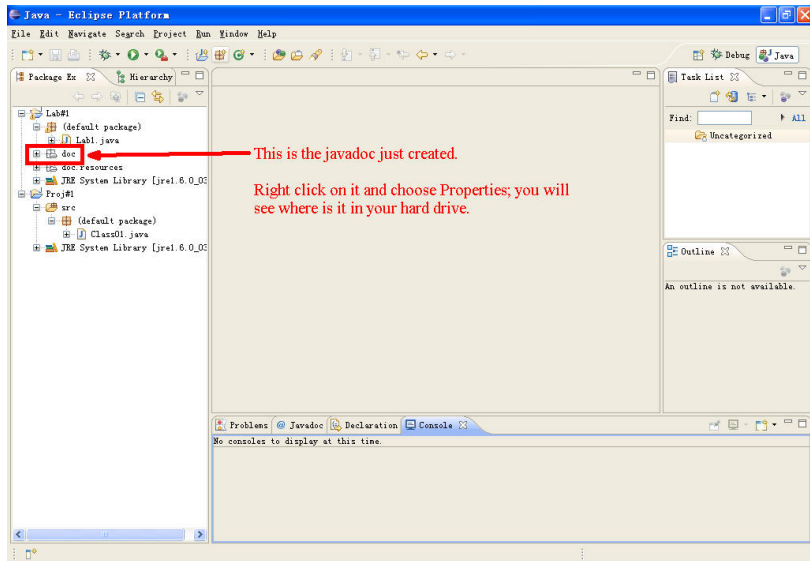
Create Javadoc (cont'd)



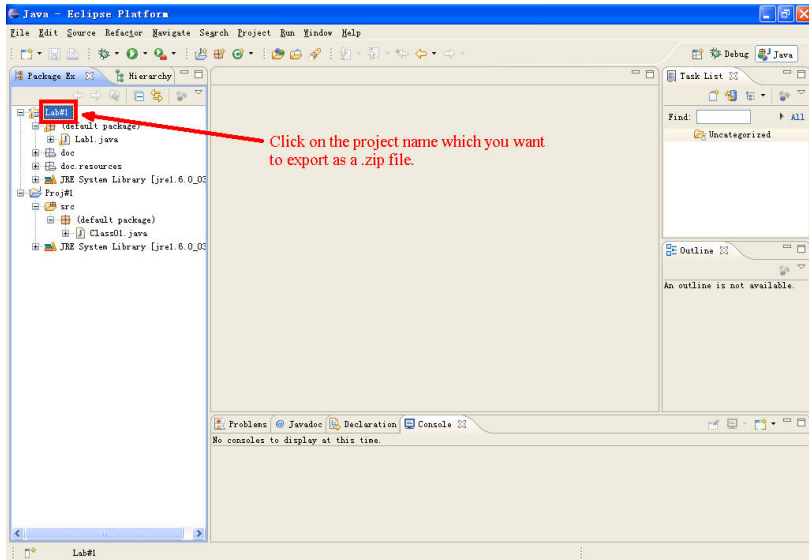
Create Javadoc (cont'd)



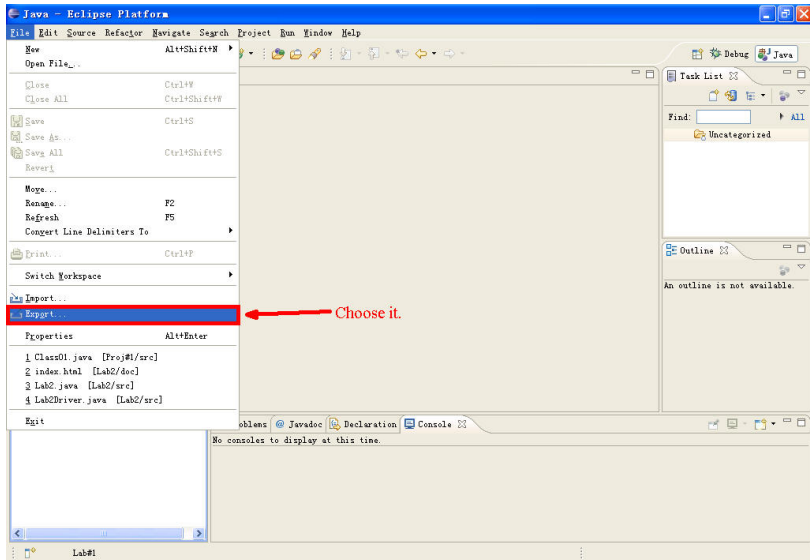
Create Javadoc (cont'd)



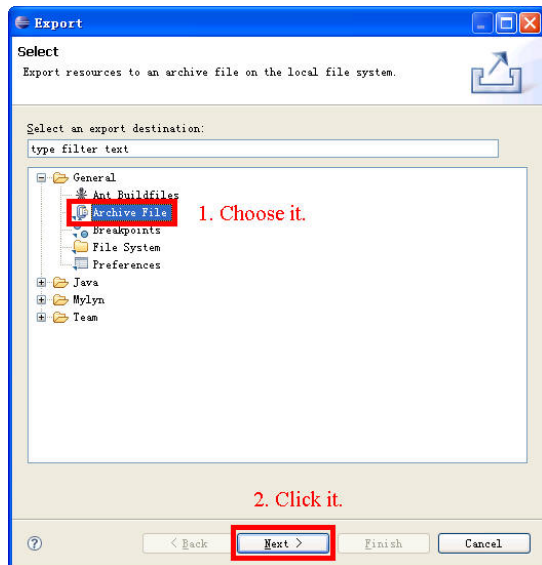
Export to a .zip File



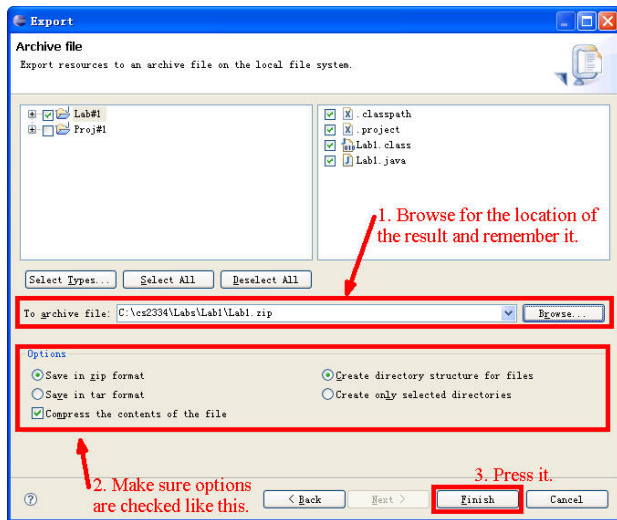
Export to a .zip File (cont'd)



Export to a .zip File (cont'd)



Export to a .zip File (cont'd)



Basic Eclipse Debugging

Breakpoints

The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows a project structure with 'Lab#2' containing 'src' and 'Lab2.java'. A red circle highlights a breakpoint icon on the 'Lab2.java' file.
- Editor:** Displays the source code of 'Lab2.java'. The code includes a class declaration, a constructor, and a 'work()' method. A red arrow points to the constructor line: `public Lab2(int iSize){`.
- Task List:** Shows 'Find:' and 'Uncategorized'.
- Outline:** Shows 'import declarations' and 'Lab2' with a tree view of methods: `myArray : int[]`, `iSize : int`, `Lab2(int)`, `init()`, `work()`, `add(int[], int)`, and `check()`.
- Console:** Shows a terminated message: `<terminated> Lab2Driver [Java Application] D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 上午10:52:51)`.

Annotations on the slide:

- A red arrow points from the text 'Make a breakpoint here. (double click)' to the breakpoint icon in the Package Explorer.
- Red text below reads: 'A breakpoint is the line where program will be paused.'

Run Eclipse Debugger

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left shows a project structure with 'Lab2Driver.java' selected and highlighted in red. A red arrow points to it with the text '1. Right click on the .java file containing the main() method.' A context menu is open over 'Lab2Driver.java', and the 'Run As' option is selected, with a sub-menu showing 'Java Application' highlighted in red. A second red arrow points to 'Java Application' with the text '2. Click it.' The main editor shows the source code of 'Lab2.java' with a 'public class Lab2' definition. The Outline view on the right shows the class structure. The console at the bottom shows the execution path: 'Program Files\Java\jre6.0.03\bin\javaw.exe (2008-1-23 上午10:52:51)'.

```
public class Lab2 {  
    /** The int array. */  
    int[] myArray;  
}
```

Run Eclipse Debugger (cont'd)

The screenshot shows the Eclipse IDE interface. A dialog box titled "Confirm Perspective Switch" is centered on the screen. The dialog contains the following text:

Confirm Perspective Switch

This kind of launch is configured to open the Debug perspective when it suspends.

This Debug perspective is designed to support application debugging. It incorporates views for displaying the debug stack, variables and breakpoint management.

Do you want to open this perspective now?

Remember my decision

Yes No

Click it.

The background shows the Eclipse IDE with the following elements:

- Package Explorer: Shows a project structure with packages Lab#1 and Lab#2. Lab#2 contains a src package with files Lab2.java and Lab2Driver.java.
- Editor: Shows the source code of Lab2.java with the following code:

```
public Lab2(int iSize) {  
    /* Assign a new size. */  
    this.iSize = iSize;  
  
    /* Initialize the array. */  
  
}
```
- Task List: Shows a search bar and a list of tasks.
- Outline: Shows a tree view of the project structure with the following items:
 - import declarations
 - Lab2
 - myArray : int[]
 - iSize : int
 - Lab2(int)
 - init()
 - work()
 - add(int[], int)
 - check()
- Console: Shows the output of the Java application: Lab2Driver [Java Application] D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 上午11:10:13)

Run Eclipse Debugger (cont'd)

Debug - Lab#2/src/Lab2.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Debug [Java Application]

Lab2Driver at localhost:4073

Thread [main] (Suspended (breakpoint at line 25 in Lab2))

Lab2.<init>(int) line: 25

Lab2Driver.main(String[]) line: 17

D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 下午07:59:27)

Name	Value
this	Lab2 (id=17)
iSize	10

Click it. You will see values of all variables defined in this class.

```
public Lab2(int iSize) {
    /* Assign a new size. */
    this.iSize = iSize;

    /* Initialize the array. */
    myArray = new int[iSize + 1];

    /* Assign some random integer to each cell of the array. */
    init();

    /* Further processing the array: the value of next cell is
```

Outline

- import declarations
 - java.util.Random
- Lab2
 - myArray : int[]
 - iSize : int
 - Lab2(int)
 - init()
 - work()
 - add(int[], int)
 - check()

Console

Lab2Driver [Java Application] D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 下午07:59:27)

Writable Smart Insert 25 : 1

Run Eclipse Debugger (cont'd)

Debug - Lab#2/src/Lab2.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Debug [Java Application]

Lab2Driver at localhost:1954

- Thread [main] (Suspended (breakpoint at line 25 in Lab2))
- Lab2.<init>(int) line: 25
- Lab2Driver.main(String[]) line: 17
- D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 上午11:10:13)

2. All variables are shown here.

Name	Value
Lab2 (id=17)	
▲ this	Lab2 (id=17)
▲ iSize	11
▲ myArray	null
○ iSize	10

```
public Lab2(int iSize) {  
    /* Assign a new size. */  
    this.iSize = iSize;  
    /* Initialize the array. */  
    myArray = new int[iSize + 1];  
    /* Assign some random integer to each cell of the array. */  
    init();  
    /* Further processing the array: the value of next cell is
```

1. Now, the program has been run to this line. That is, all codes in this file above this line have been executed. This line is not yet executed but it will be if "Step Into" or "Step Over" button is pressed.

Console [Java Application] D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 上午11:10:13)

Writable Smart Insert 25 : 1

2008年1月23日

Useful Buttons

The screenshot shows the Eclipse IDE's Debug console. The title bar reads "Debug - Lab#2/src/Lab2.java - Eclipse Platform". The menu bar includes "File", "Edit", "Source", "Refactor", "Navigate", "Search", "Project", "Run", "Window", and "Help". The toolbar contains various icons for debugging. Three buttons are highlighted with red boxes and red arrows pointing to them from text labels above:

- "Jump to the next breakpoint." points to the "Next Breakpoint" button (a green play button with a right-pointing arrow).
- "Terminate the debugger." points to the "Terminate" button (a red square button).
- "Step Into." points to the "Step Into" button (a blue play button with a downward-pointing arrow).

Below the toolbar, the Debug console shows the following structure:

- Lab2Driver [Java Application]
- Lab2Driver at localhost:4073
 - Thread [main] (Suspended (breakpoint at line 25 in Lab2))
 - Lab2.<init>(int) line: 25
 - Lab2Driver.main(String[]) line: 17

At the bottom of the console, the command prompt shows: "D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 下午07:59:27)".

Below the console, the text "Step Over." is written in red, with a red arrow pointing to the "Step Over" button (a blue play button with an upward-pointing arrow) in the toolbar.

Step Over

Debug - Lab#2/src/Lab2.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

The "Step Over" button.

Debug [Java Application]

- Lab2Driver at localhost:1954
 - Thread [main] (Suspended (breakpoint at line 25 in Lab2))
 - Lab2.<init>(int) line: 25
 - Lab2Driver.main(String[]) line: 17

D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 上午11:10:13)

Name	Value
this	Lab2 (id=17)
iSize	11
myArray	null
iSize	10

```
public Lab2(int iSize) {  
    /* Assign a new size. */  
    this.iSize = iSize;  
  
    /* Initialize the array. */  
    myArray = new int[iSize + 1];  
  
    /* Assign some random integer to each cell of the array. */  
    init();  
  
    /* Further processing the array: the value of next cell is...
```

Outline

- import declarations
- Lab2
 - myArray : int[]
 - iSize : int
 - Lab2(int)
 - init()
 - work()
 - add(int[], int)
 - check()

Console

Lab2Driver [Java Application] D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 上午11:10:13)

To move to here, there are two ways:
1. Click twice the "Step Over" button.
2. Right click the destination line -> Run to Line.

Writable Smart Insert 25 : 1

2008年1月23日

Variables

The screenshot shows the Eclipse IDE in a debug state. The main editor displays the source code for `Lab2.java`. The `myArray` variable is highlighted in green, indicating it is the current focus of the inspection. The `Variables` window on the right shows the current state of the variables in the scope. The `myArray` variable is highlighted in yellow, and its value is shown as `int[11] (id=18)`. A red arrow points from the `myArray` variable in the code to its entry in the `Variables` window.

Once the value of a variable is changed, it will be highlighted.

```
public Lab2(int iSize) {
    /* Assign a new size. */
    this.iSize = iSize;

    /* Initialize the array. */
    myArray = new int[iSize + 1];

    /* Assign some random integer to each cell of the array. */
    init();

    /* Further processing the array: the value of next cell is
    * the sum of its current value and its previous neighbor's
    */
}
```

Name	Value
this	Lab2 (id=17)
iSize	10
myArray	int[11] (id=18)
iSize	10

Step Into

The screenshot shows the Eclipse IDE in a debug state. The top toolbar contains the 'Step Into' button, which is highlighted with a red box and a red arrow pointing to it. A red text label 'The "Step Into" button.' points to this button. The 'Debug' console on the left shows the execution stack, with 'Lab2Driver.main(String[]) line: 17' selected. A red arrow points from this stack entry to the 'init()' method in the 'Lab2.java' editor. A red text label 'We are here. Press "Step Into" button, then we can goto the definition of init() method.' points to this arrow. The 'Variables' view on the right shows the current state of variables: 'this' (Lab2 (id=17)), 'iSize' (10), 'myArray' (int[11] (id=18)), and 'iSize' (10). The 'Outline' view on the right shows the class structure, with 'init()' selected. The 'Console' view at the bottom shows the execution path: 'Lab2Driver [Java Application] D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 上午11:29:48)'. The status bar at the bottom indicates 'Writable Smart Insert 31 : 1'.

Debug - Lab#2/src/Lab2.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Debug [Java Application]

Lab2Driver at localhost:2083

Thread [main] (Suspended)

Lab2.<init>(int) line: 31

Lab2Driver.main(String[]) line: 17

D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 上午11:29:48)

The "Step Into" button.

We are here. Press "Step Into" button, then we can goto the definition of init() method.

00- Variables

Name	Value
this	Lab2 (id=17)
iSize	10
myArray	int[11] (id=18)
iSize	10

```
/* assign some random integer to each cell of the array. */
init();

/* Further processing the array: the value of next cell is
 * the sum of its current value and its previous neighbor's.
 */
work();

/* Perform some calculation. */
check();
}
```

import declarations

- Lab2
 - myArray : int[]
 - iSize : int
 - Lab2(int)
 - init()
 - work()
 - add(int[], int)
 - check()

Console

Lab2Driver [Java Application] D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 上午11:29:48)

Writable Smart Insert 31 : 1

Step Into (cont'd)

Debug - Lab#2/src/Lab2.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Debug [Java Application]

- Lab2Driver at localhost:2983
 - Thread [main] (Suspended)
 - Lab2.init() line: 48
 - Lab2.<init>(int) line: 31
 - Lab2Driver.main(String[]) line: 17

D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 上午11:29:48)

Name	Value
this	Lab2 (id=17)

```
/**
 * Assign some random integer to each cell of the array.
 */
public void init() {
    Random rand = new Random();
    for (int i = 0; i <= iSize; i++) {
        myArray[i] = rand.nextInt() % 10;
    }
}
```

we step into the init() method.

import declarations

- Lab2
 - myArray : int[]
 - iSize : int
 - Lab2(int)
 - init()
 - work()
 - add(int[], int)
 - check()

Lab2Driver [Java Application] D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 上午11:29:48)

Conditional Breakpoints

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

- Top Panel:** Eclipse Platform title bar and menu bar (File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help).
- Debug Console:** Shows the execution stack for Lab2Driver, including Thread [main] (Suspended) and the current line of code: Lab2Driver.main(String[]) line: 17.
- Variables View:** Shows a variable 'this' with value 'Lab2 (id=17)'. The 'Breakpoints' tab is also visible but empty.
- Editor:** Displays the source code of Lab2Driver.java. A red box highlights the line `myArray[i] = rand.nextInt() % 10;` inside a for loop. A red arrow points to this line from the text 'First, make a breakpoint.'.
- Outline View:** Shows the class structure with methods: `import declarations`, `Lab2`, `myArray : int[]`, `iSize : int`, `Lab2(int)`, `init()`, `work()`, `add(int[], int)`, and `check()`.
- Console:** Shows the output: 'Lab2Driver [Java Application] D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 下午12:16:25)'. The status bar at the bottom indicates 'Writable', 'Smart Insert', and '46 : 1'.

Suppose we want the program paused in the third iteration. How to do it?

First, make a breakpoint.

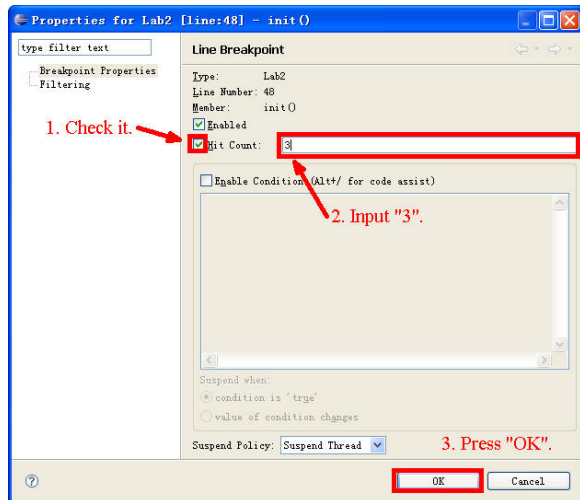
Conditional Breakpoints (cont'd)

The screenshot shows the Eclipse IDE interface. The main editor displays the source code for `Lab2Driver.java`. A breakpoint is set on the line `Random rand = new Random();`. A context menu is open over this breakpoint, and the option `Breakpoint Properties...` is highlighted with a red box. A red arrow points from the text "Right click on the breakpoint, then select this." to the highlighted option.

The IDE also shows the following components:

- Debug Console:** Shows the execution stack, including `Lab2Driver` at `localhost:1421` and `Thread [main] (Suspended)`.
- Variables View:** Shows a variable `this` with value `Lab2 (id=17)`.
- Outline View:** Shows the class structure, including `import declarations`, `Lab2`, `myArray : int[]`, `iSize : int`, `Lab2(int)`, `init()`, `work()`, `add(int[], int)`, and `check()`.

Conditional Breakpoints (cont'd)



Conditional Breakpoints (cont'd)

1. Press "Resume".

2. The program paused at the third iteration.

```
Debug - Lab#2/src/Lab2.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help

Debug [Java Application]
  Lab2Driver [Java Application]
    Lab2Driver at localhost:2303
      Thread [main] Suspended (breakpoint at line 48 in Lab2)
        Lab2.init() line: 48
        Lab2.init()>(int) line: 31
        LabDriver.main(String[]) line: 17
        D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 下午12:34:43)

00- Variables
Name      Value
this     Lab2 (id=17)
rand     Random (id=18)
i        2

Lab2Driver.java | Lab2.java
/**
 * Assign some random integer to each cell of the array.
 */
public void init() {
    Random rand = new Random();
    for (int i = 0; i <= iSize; i++) {
        myArray[i] = rand.nextInt() % 10;
    }
}
/**
 * Perform addition on each cell of myarray
 */

import declarations
Lab2
  myArray : int[]
  iSize : int
  Lab2(int)
  init()
  work()
  add(int[], int)
  check()

Lab2Driver [Java Application] D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 下午12:34:43)
```

Terminate debugger

Debug - Lab#2/src/Lab2.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Debug [Java Application]

Lab2Driver at localhost:2303

- Thread [main] (Suspended (breakpoint at line 48 in Lab2))
- Lab2.init() line: 48
- Lab2.<init>(int) line: 31
- Lab2Driver.main(String[]) line: 17

D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 下午12:34:43)

00- Variables

Name	Value
this	Lab2 (id=17)
rand	Random (id=18)
i	2

Breakpoints

Lab2Driver.java | Lab2.java

```
/**
 * Assign some random integer to each cell of the array.
 */
public void init() {
    Random rand = new Random();
    for (int i = 0; i <= iSize; i++) {
        myArray[i] = rand.nextInt() % 10;
    }
}
/**
 * Perform addition on each cell of myArray
```

Outline

- import declarations
- Lab2
 - myArray : int[]
 - iSize : int
 - Lab2(int)
 - init()
 - work()
 - add(int[], int)
 - check()

Console

Lab2Driver [Java Application] D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 下午12:34:43)

Press this or that button.

Questions?