

CS2334 Lab2

Eclipse And Debugging Survival Guide

Yu-Hsin Li and Mark Woehrer

Fall 2008

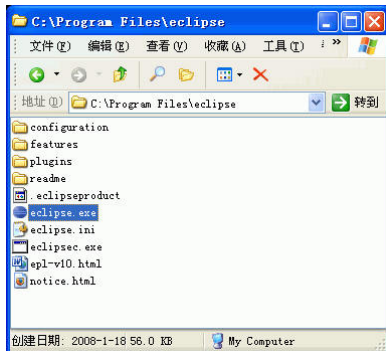
Basic Eclipse Tutorial

Basic Eclipse Debugging

Basic Eclipse Tutorial

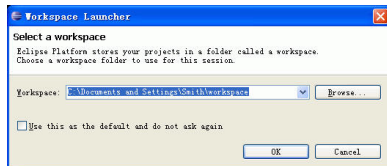
Start

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

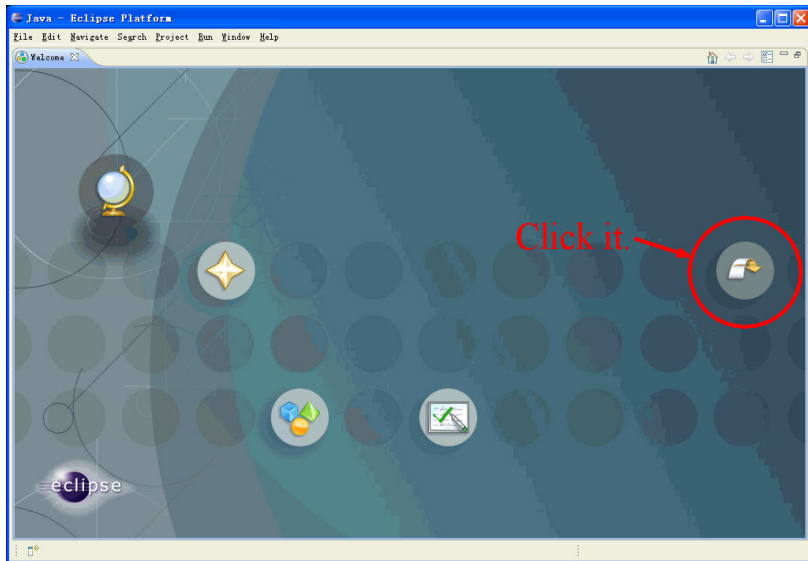


Choose a Workspace

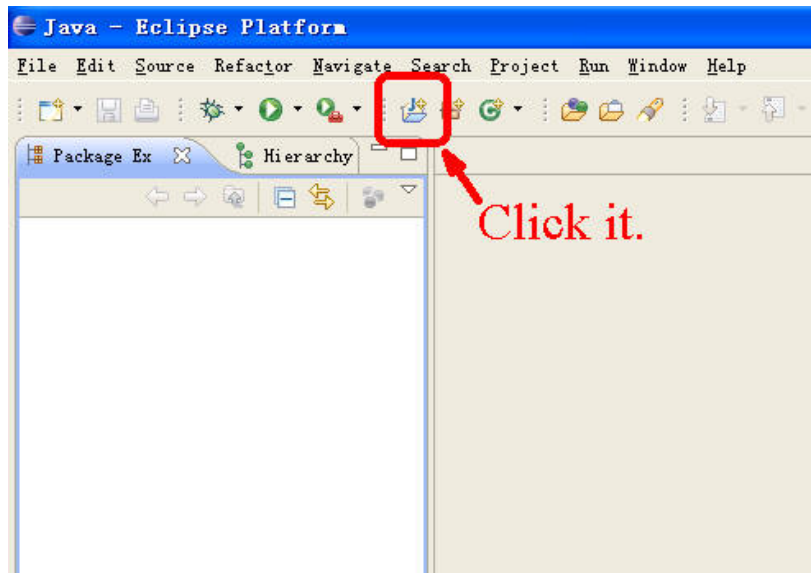
- ▶ A workspace is a folder/directory in your hard drive.
- ▶ Each workspace houses a collection of projects.
- ▶ Click OK for the default one.



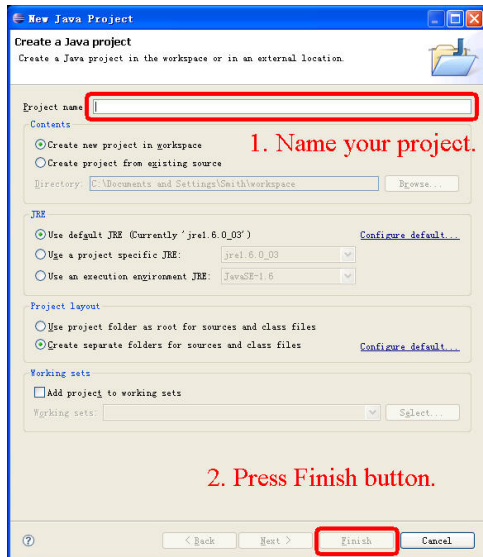
The Welcome Screen



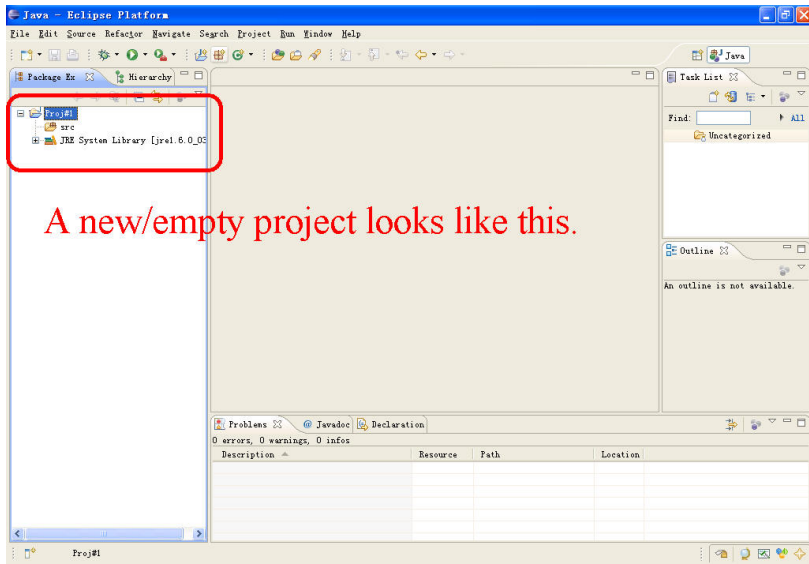
Create a New Project



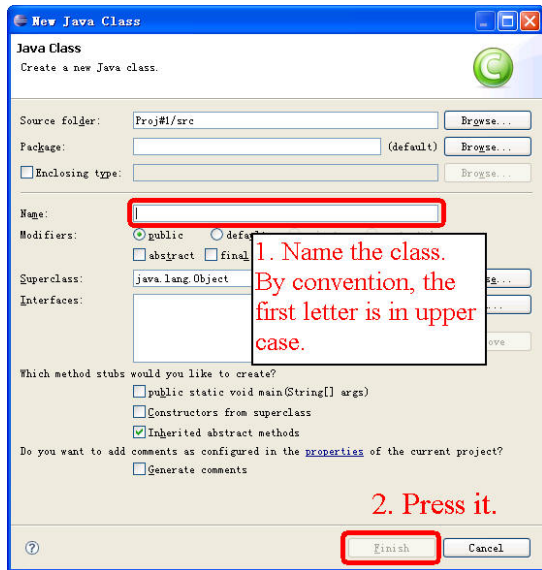
Create a New Project (Cont'd)



Create a New Project (Cont'd)



Add a Class to a Project (Cont'd)



Add a Class to a Project (Cont'd)

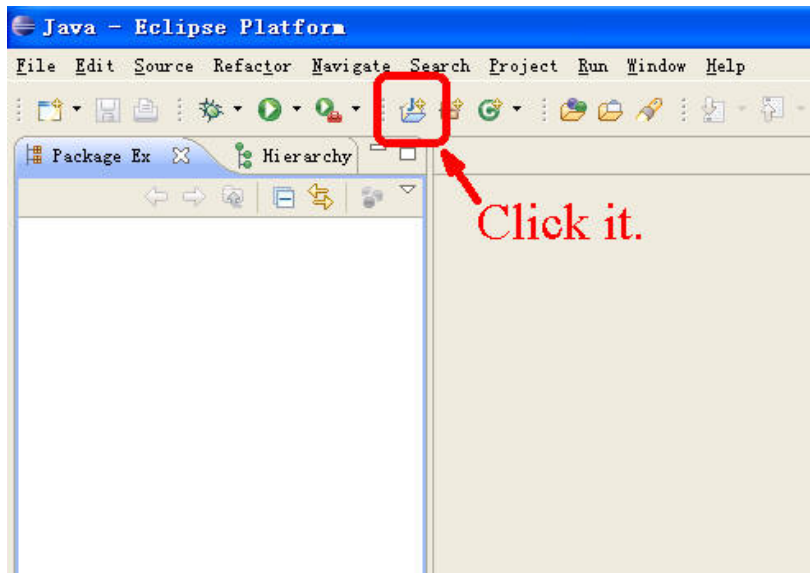
The screenshot shows the Eclipse IDE interface. The Package Explorer on the left shows a project named 'Proj#1' with a source folder 'src' containing a class 'Class01.java', which is highlighted with a red box. The main editor window shows the code for 'Class01.java' with the following content:

```
public class Class01 {  
  
}
```

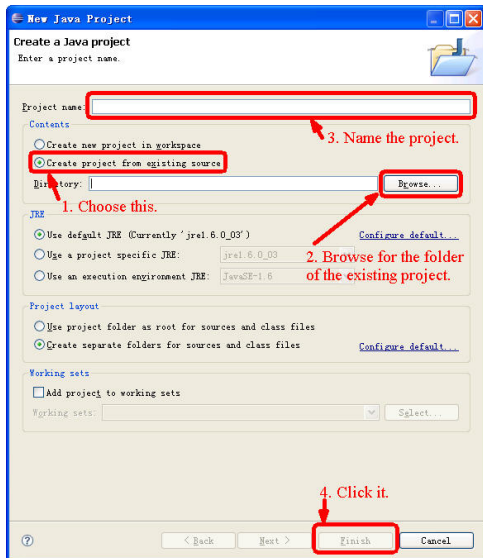
Below the code editor, the Outline view shows 'Class01'. The Problems view at the bottom shows '0 errors, 0 warnings, 0 infos'. A red text box is overlaid on the screenshot with the following text:

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.

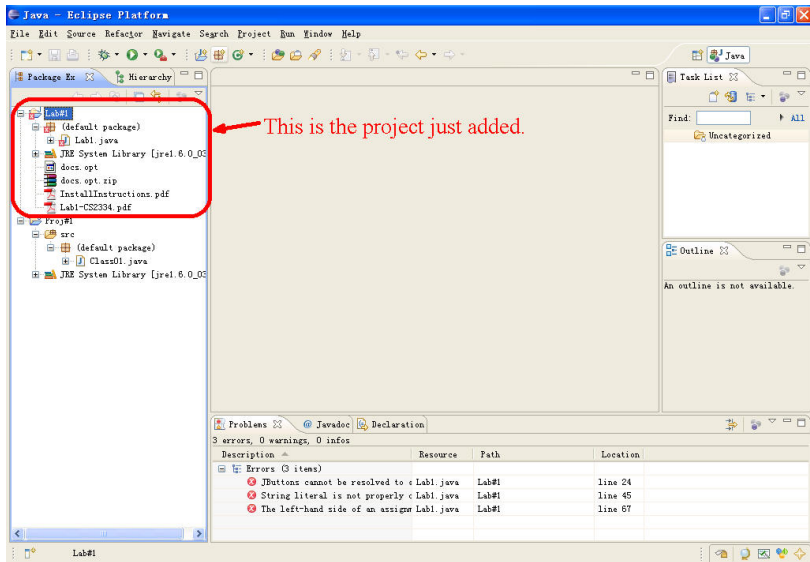
Add a Directory As a New Project



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



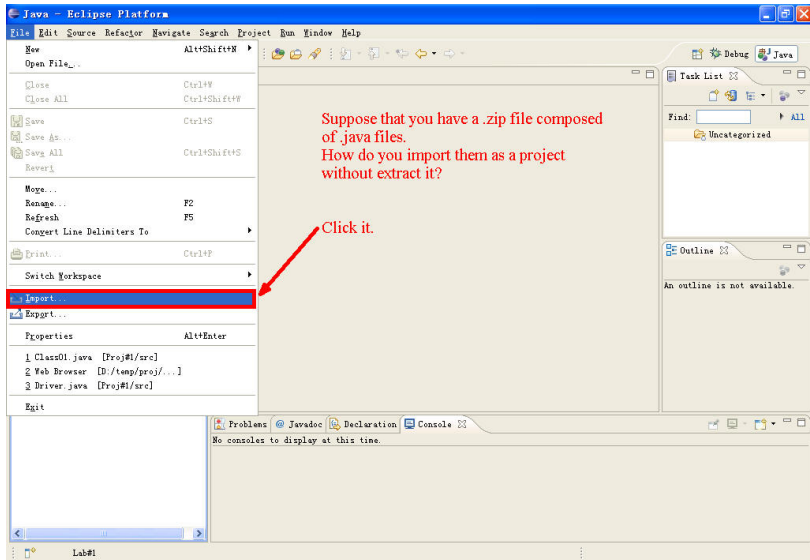
Add a Directory As a New Project (Cont'd)



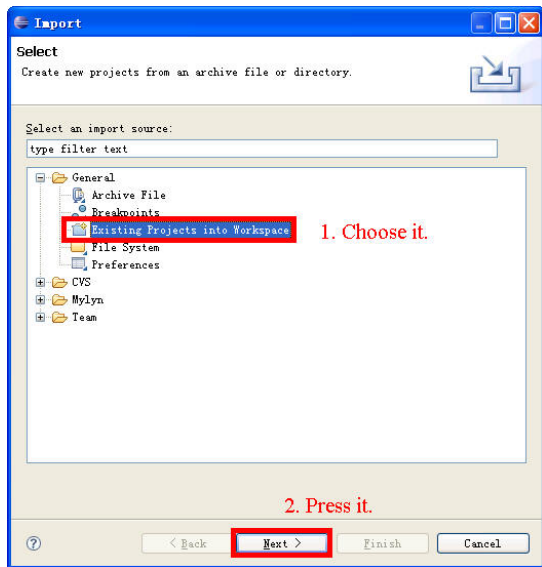
The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project named 'Lab#1' which is highlighted with a red box. A red arrow points to this box with the text 'This is the project just added.' The project structure includes a default package, Lab#1.java, and the JRE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

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

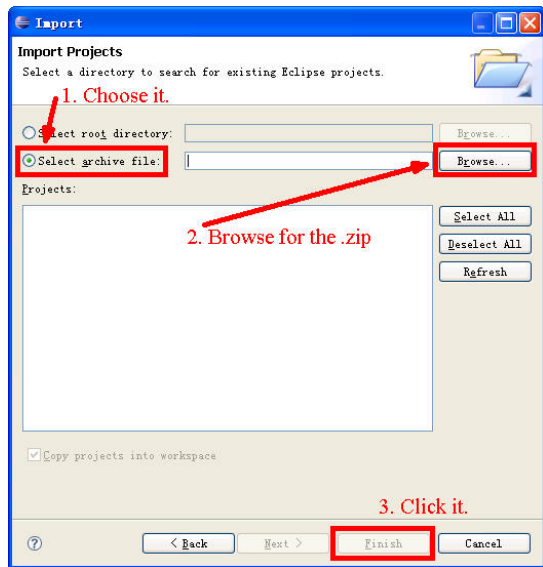
Add a .zip File As a New Project



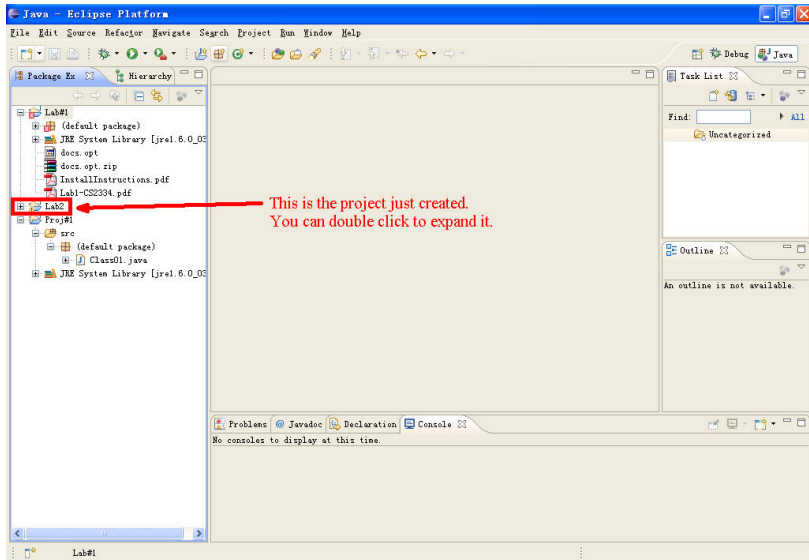
Add a .zip File As a New Project (Cont'd)



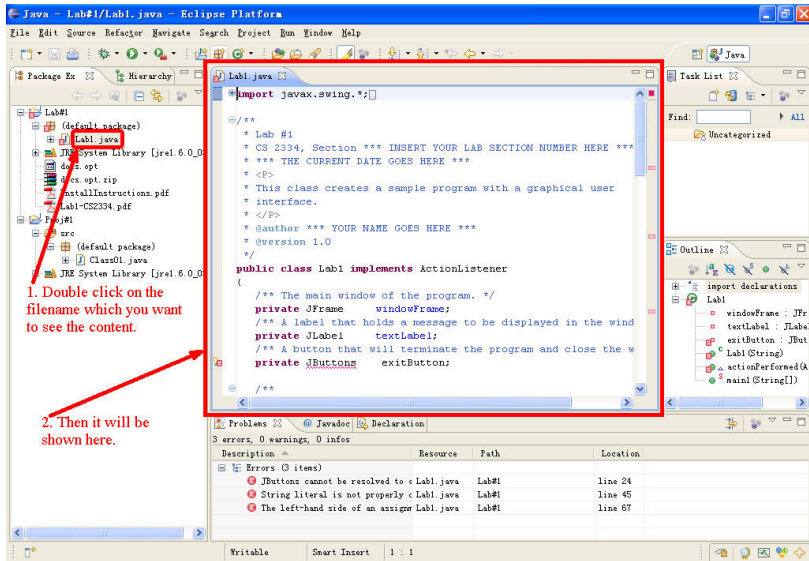
Add a .zip File As a New Project (Cont'd)



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



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)			
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 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:

- Line 24: `JButtons` cannot be resolved to a type.
- Line 45: A string literal is not properly enclosed in quotes.
- Line 67: The left-hand side of an assignment is not a variable.

 The `Problems` view at the bottom shows these errors in a table:

Description	Resource	Path	Location
JButtons cannot be resolved to a type.	Lab1.java	Lab#1	line 24
String literal is not properly enclosed in quotes.	Lab1.java	Lab#1	line 45
The left-hand side of an assignment is not a variable.	Lab1.java	Lab#1	line 67

Annotations in the image include:

- A red circle around the Package Explorer icon with the text: "Move mouse over here, Eclipse will pop up the error message."
- Red circles around the error icons in the code editor with the text: "These are indices of errors. You can click on it."
- A red circle around the Problems view icon with the text: "All errors are shown here."

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, including an unresolved type `JButtons`. A red circle highlights the `JButtons` type in the code, and a red arrow points to it from the text below. A red box highlights the quick fix menu that appears, listing several options to resolve the error.

Click it. Eclipse will show possible solutions for you.

```

/** 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;
private JButtons buttons;

public Lab1() {
    // Create the main window.
    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" );
}
    
```

Quick Fix Menu Options:

- Create class 'JButtons'
- Create interface 'JButtons'
- Change to 'JButton' (javax.swing)
- Create enum 'JButtons'
- Add type parameter 'JButtons' to 'Lab1'
- Rename in file (Ctrl+R, R direct access)

Problems List:

Description	Resource	Path	Location
JButtons cannot be resolved to a type	Lab1.java	Lab#1	line 24
String literal is not properly enclosed in quotes	Lab1.java	Lab#1	line 45
The left-hand side of an assignment must be a variable or a field name	Lab1.java	Lab#1	line 67

Compile a .java File

The screenshot shows the Eclipse IDE interface. The 'Project' menu is open, and 'Build Automatically' is checked and highlighted with a red box. A red arrow points from the text below to this menu item. The main editor shows the source code for 'Lab1.java'. The Problems view at the bottom shows three errors:

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 Lab1.java	Lab#1		line 67

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.

Compile a .java File (cont'd)

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

```
/** 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;

/**
 * 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 th
 */
public Lab1( String message )
{
    // Create a text label for the program.
    textLabel = new JLabel( message );

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

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

How to Run a Project

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

2. Choose this.

The screenshot shows the Eclipse IDE with the following elements:

- Package Explorer:** Shows a project named 'Lab#1' with a sub-package 'default package' containing the file 'Lab1.java'. A red box highlights 'Lab1.java'.
- Editor:** Displays the source code of 'Lab1.java'. The code includes an import statement for 'javax.swing.*', a comment indicating where to insert a lab section number, and a class definition for 'Lab #1' that implements 'ActionListener' and contains a 'main' method.
- Context Menu:** A right-click context menu is open over 'Lab1.java'. The 'Run As' option is highlighted in blue, and a red box highlights it with the text '2. Choose this.'.
- Task List:** Shows 'Uncategorized'.
- Outline:** Shows the class structure with 'main(String[])' listed as a method.
- Declarations Table:** Shows a table with columns for 'Name' and 'Location'. The entry 'Lab #1 Java Application' is highlighted in red, with 'Alt+Shift+X, J' listed as the keyboard shortcut and 'line 85' as the location.

Run a Project with Command Line Arguments

The screenshot shows the Eclipse IDE interface. The Package Explorer on the left shows a project named 'Lab#1' with a sub-package 'src' containing a file 'Lab1.java'. A right-click context menu is open over 'Lab1.java'. The 'Run As' option is selected, and the 'Open Run Dialog...' sub-option is highlighted with a red box and an arrow. A red arrow also points to 'Lab1.java' in the Package Explorer with the text '1. Right click on the .java file containing the main() method.' Another red arrow points to the 'Open Run Dialog...' option with the text '2. Click this.'

The main editor window shows the source code of 'Lab1.java' with the following content:

```

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

// sample program with a graphical user
// interface.

// *** GOES HERE ***

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

public class Lab1 implements ActionListener
{
    // of the program. */
    private JFrame windowFrame;
    private JLabel textLabel;
    private JButton exitButton;
    private JButton actionPerformed(A
    private Lab1(String[] args);
    private void init();
    private void run();
    private void terminate();
}

```

The Outline view on the right shows the class structure:

```

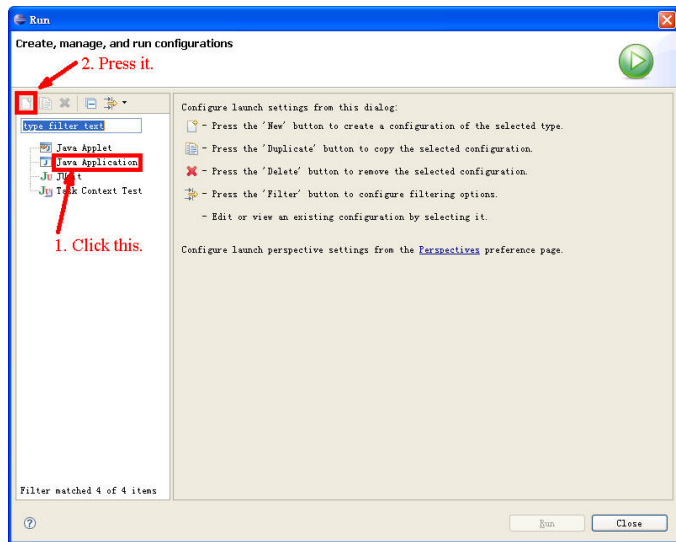
import declarations:
- Lab1
  - windowFrame : JFrame
  - textLabel : JLabel
  - exitButton : JButton
  - Lab1(String)
  - actionPerformed(A

```

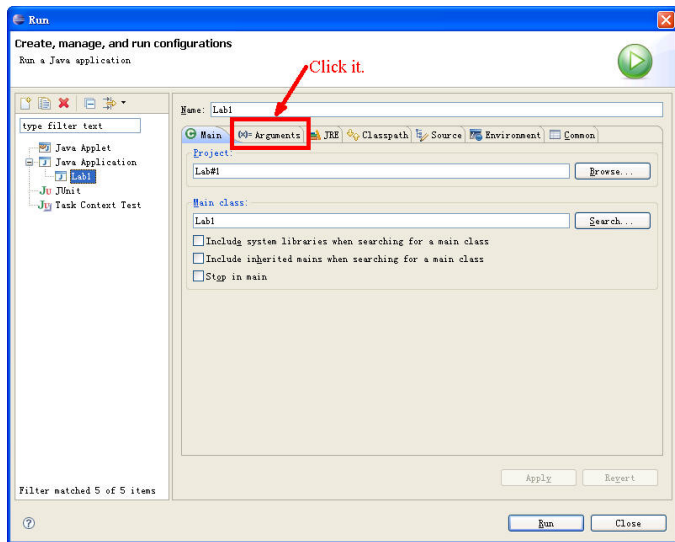
The Run As dialog box is open, showing the following table:

Resource	Path	Location
Java Application	Alt+Shift+X, J	line 85
Open Run Dialog...		

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



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



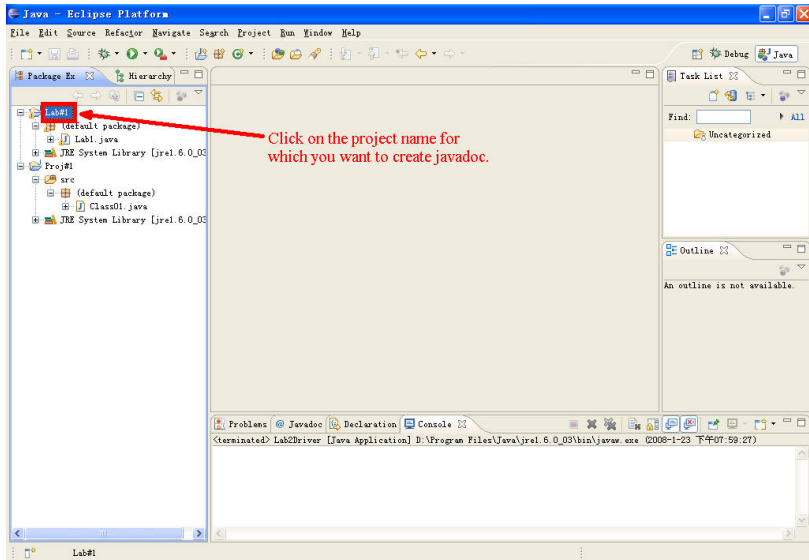
Run a Project with Command Line Arguments (cont'd)

The screenshot shows the Eclipse IDE's 'Run' dialog box. The title bar reads 'Run' and the subtitle is 'Create, manage, and run configurations'. Below the subtitle, it says 'Run a Java application'. On the left, there is a tree view of project configurations including 'Java Applet', 'Java Application', 'Lab1', 'JUnit', and 'Task Context Test'. The main area is titled 'Name: Lab1' and contains several tabs: 'Main', 'Arguments', 'JRE', 'Classpath', 'Source', 'Environment', and 'Common'. The 'Arguments' tab is active, showing a text area for 'Program arguments:' containing the text 'input1.txt input2.txt'. A red arrow points from the text '1. Place all arguments here.' to this text area. Below this are fields for 'VM arguments:' and 'Working directory:'. At the bottom right, there are buttons for 'Apply', 'Revert', 'Run', and 'Close'. A red arrow points from the text '2. Click it.' to the 'Run' button.

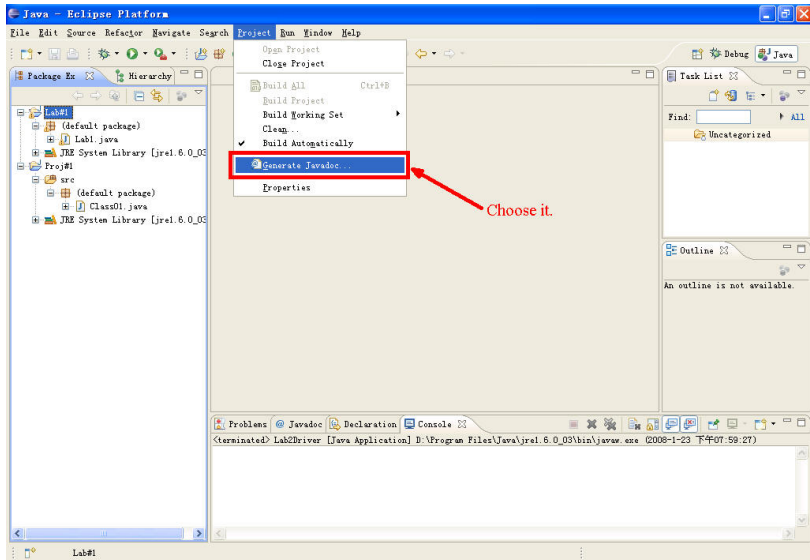
1. Place all arguments here.

2. Click it.

Create Javadoc



Create Javadoc (cont'd)



Create Javadoc (cont'd)

The screenshot shows the 'Generate Javadoc' dialog box in Eclipse. It has a title bar 'Generate Javadoc' and a 'Javadoc Generation' section. The 'Javadoc command:' field contains 'D:\Program Files\Java\jdk1.6.0_03\bin\javadoc.exe' and is highlighted with a red box. A red arrow points to this field with the text '1. According to your computer, input the correct path for javadoc.exe'. Below this, there are two tree views for selecting types to generate Javadoc for. The first tree view contains 'Lab#1' and 'Proj#1', with 'Lab#1' selected and highlighted by a red box. A red arrow points to this box with the text '2. Choose it.'. The second tree view is empty. A red arrow points to the right side of the dialog with the text '3. This will be the folder containing javadoc files.'. Under 'Create Javadoc for members with visibility:', the 'Private' radio button is selected and highlighted with a red box. Below this, the 'Use Standard Doclet' radio button is selected. The 'Destination:' field contains 'D:\temp\proj\workspace\cs2334 lab1\doc' and is highlighted with a red box. At the bottom, the 'Next >' button is highlighted with a red box. A red arrow points to this button with the text '4. Press it.'. The dialog also includes a 'Configure...' button, a 'Browse...' button, and a '?' icon.

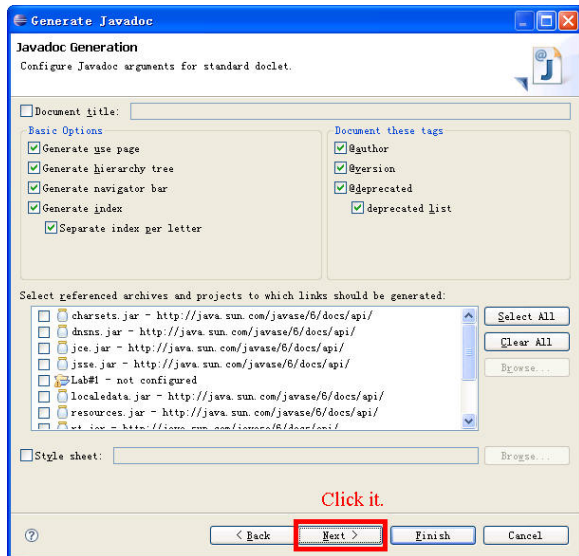
1. According to your computer, input the correct path for javadoc.exe

2. Choose it.

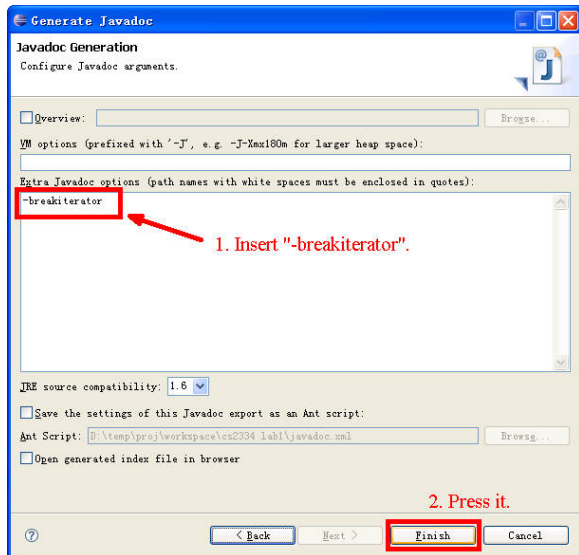
3. This will be the folder containing javadoc files.

4. Press it.

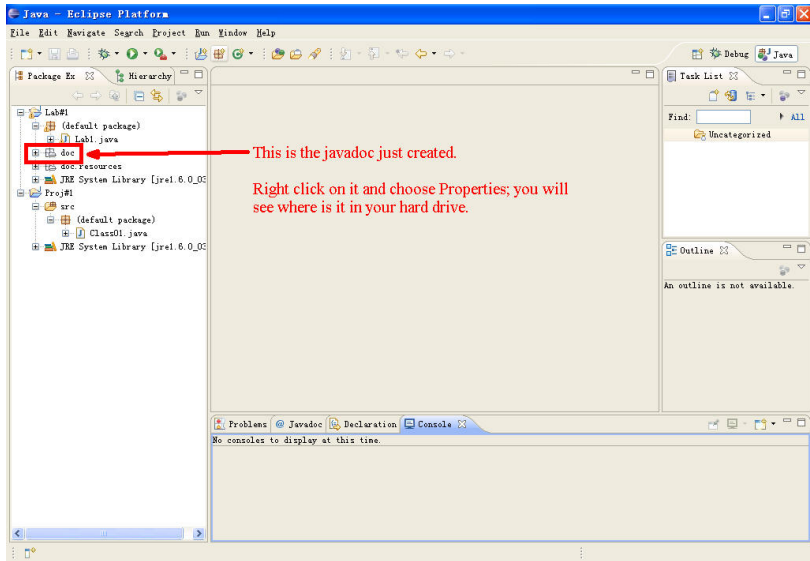
Create Javadoc (cont'd)



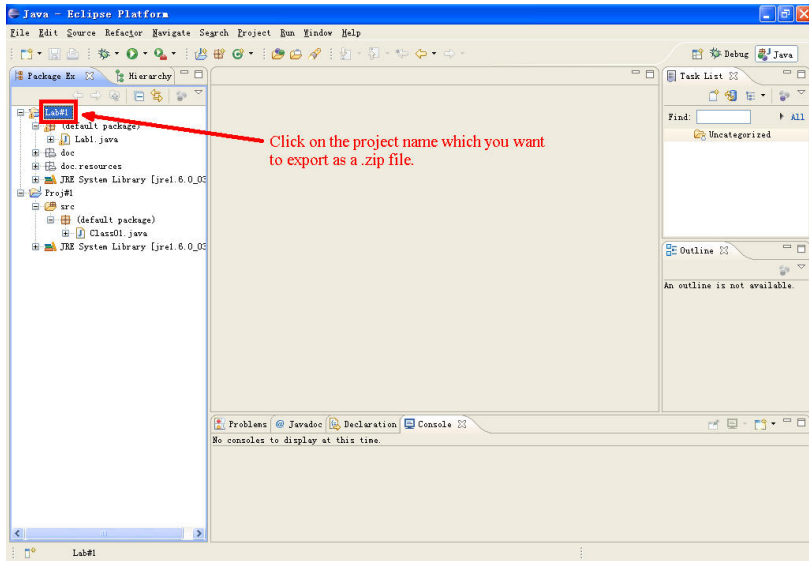
Create Javadoc (cont'd)



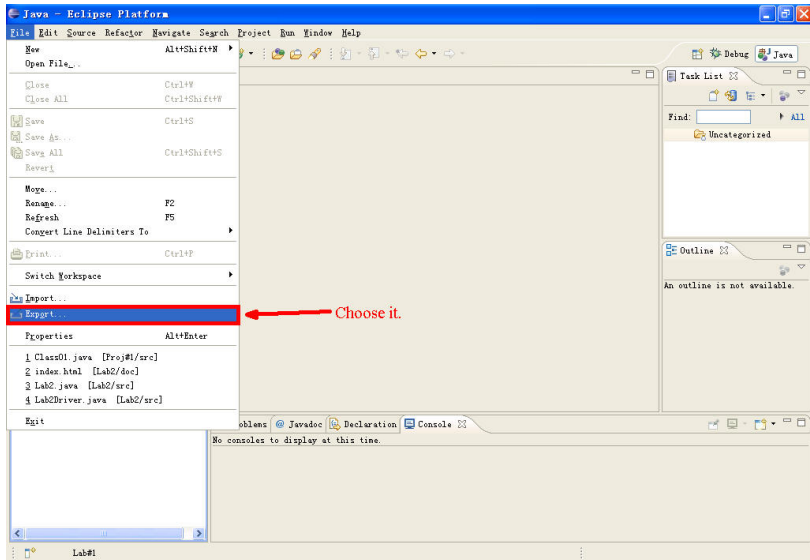
Create Javadoc (cont'd)



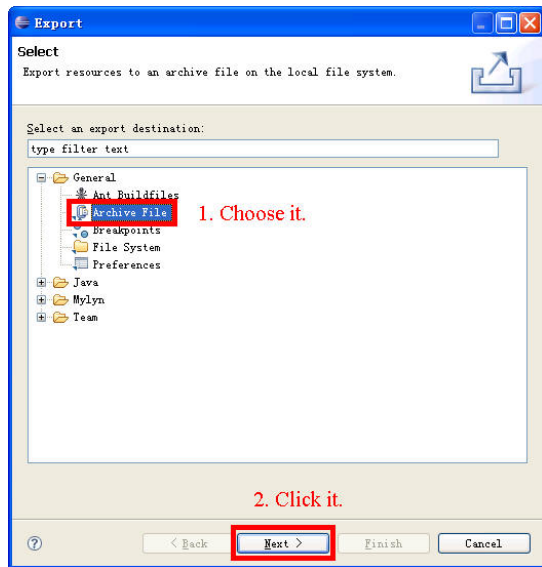
Export to a .zip File



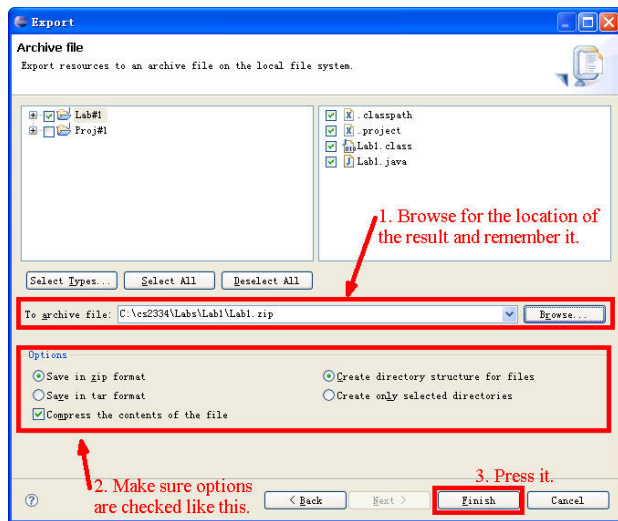
Export to a .zip File (cont'd)



Export to a .zip File (cont'd)



Export to a .zip File (cont'd)



Basic Eclipse Debugging

Breakpoints

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

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Hierarchy

Lab#1 (default package)
 Lab1.java
 JRE System Library [jre1.6.0_03]
 Lab#2 (default package)
 src (default package)
 Lab2.java
 Lab2Driver.java
 JRE System Library [jre1.6.0_03]

```

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

    /** The size of array. */
    int iSize = 11;

    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
        * the sum of its current value and its previous neighbor
        */
        work();
    }
  }

```

Task List
 Find: All
 Uncategorized

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

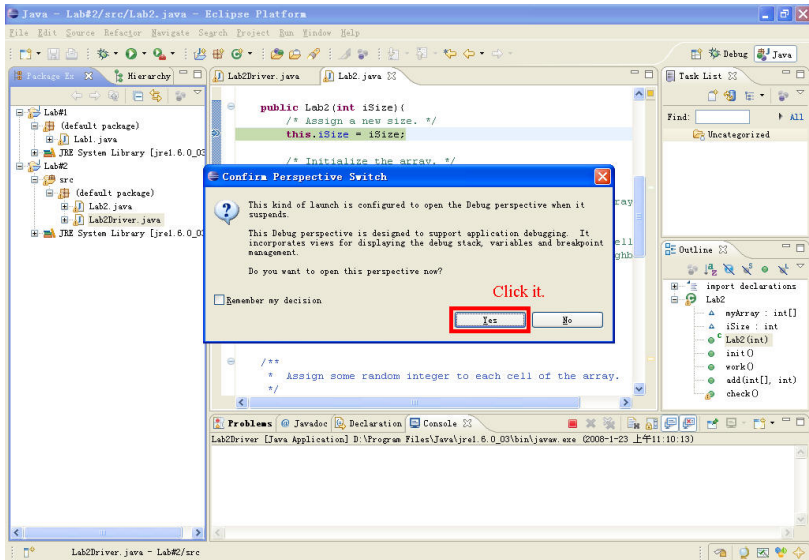
Problems Javadoc Declaration Console
 <terminated> Lab2Driver [Java Application] D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 上午10:52:51)

Writable Smart Insert 23 : 1

Make a breakpoint here. (double click)

A breakpoint is the line where program will be paused.

Run Eclipse Debugger (cont'd)



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
```

import declarations
java.util.Random
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07:59:27)

Writable Smart Insert 25 : 1

Run Eclipse Debugger (cont'd)

2. All variables are shown here.

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 call is

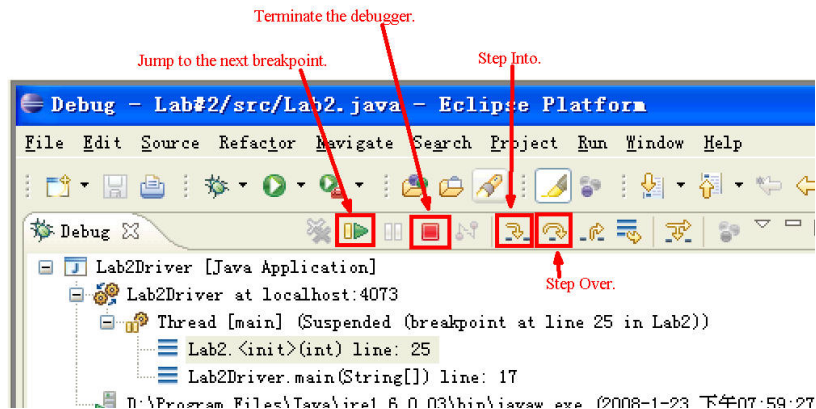
```

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

2008年1月23日

Useful Buttons



Step Over

The "Step Over" button.

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

```

public Lab2(int iSize) {
    /* Initialize 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();

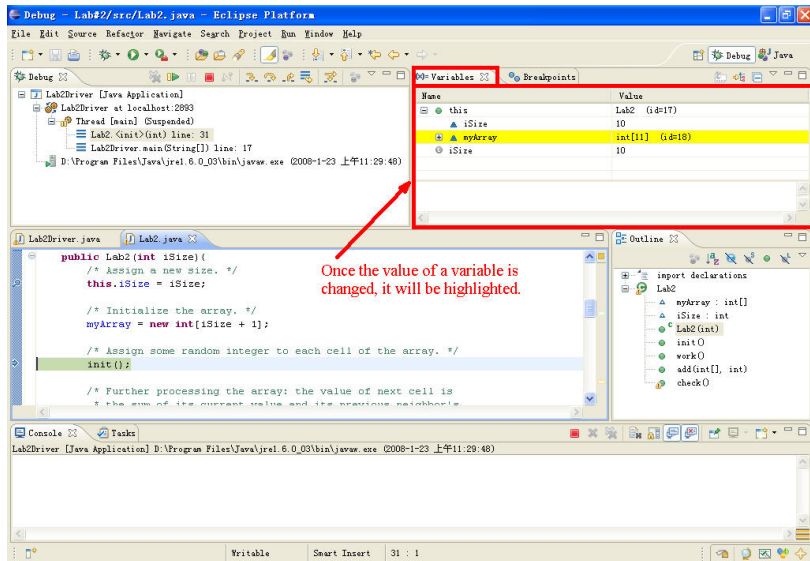
    /* Further processing the array: the value of next call is
  
```

We are here.

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

Console: Lab2Driver [Java Application] D:\Program Files\Java\jre6.0.03\bin\javaw.exe (2008-1-23 上午11:10:13)

Variables



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

File Edit Source Refactor Navigate Search Project Run Window Help

Debug [Java Application]

Lab2Driver at localhost:2893

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)

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

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
    value. */
}
```

import declarations

Lab2

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

Console [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

The "Step Into" button.

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

```
/* 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()

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

Step Into (cont'd)

The screenshot shows the Eclipse IDE in a debug state. The top toolbar includes buttons for File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The 'Debug Console' shows the execution flow of the application, with the following stack trace:

```
Lab2Driver [Java Application]
├── Lab2Driver at localhost:2893
│   ├── Thread [main] (Suspended)
│   │   ├── Lab2.init() line: 46
│   │   ├── 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 'Variables' view shows the current state of the program:

Name	Value
this	Lab2 (id=17)

The 'Lab2Driver.java' file is open, and the 'init()' method is highlighted with a red box. The code is as follows:

```
/**
 * 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.
```

The 'Outline' view shows the class structure:

```
import declarations
├── Lab2
│   ├── myArray : int[]
│   ├── iSize : int
│   ├── Lab2 (int)
│   ├── init ()
│   ├── work ()
│   ├── add (int[], int)
│   └── check ()
```

A red arrow points to the 'init ()' method in the Outline view. A red box highlights the 'init()' method in the source code. A red arrow points to the 'init()' method in the Outline view. A red box highlights the 'init()' method in the source code. A red arrow points to the 'init()' method in the Outline view.

We step into the init() method.

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

Conditional Breakpoints (cont'd)

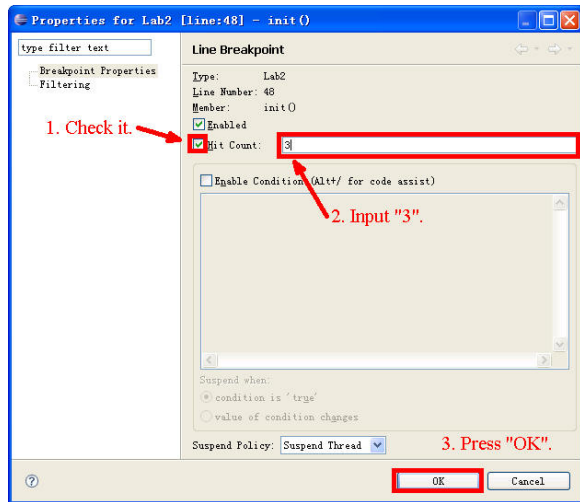
The screenshot shows the Eclipse IDE in a debug state. The main editor displays the source code of `Lab2.java` with a breakpoint set on the line `Random rand = new Random();`. A context menu is open over this breakpoint, and the option `Breakpoint Properties...` is highlighted with a red box. A red arrow points from the text "Right click on the breakpoint, then select this." to the highlighted option.

The IDE interface includes the following components:

- Top Bar:** Title bar "Debug - Lab#2/src/Lab2.java - Eclipse Platform" and menu bar (File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help).
- Toolbar:** Standard Eclipse development icons.
- Left Panel (Debug Console):** Shows the execution stack with `Lab2Driver` at `localhost:1421` and `Thread [main] (Suspended)`. The stack trace includes `Lab2.init() line: 48`, `Lab2.<init>(int) line: 31`, and `Lab2Driver.main(String[]) line: 17`.
- Right Panel (Variables):** A table showing the current state of variables:

Name	Value
this	Lab2 (id=17)
- Bottom Panel (Outline):** Shows the class structure for `Lab2`, including `import declarations`, `myArray : int[]`, `iSize : int`, `Lab2 (int)`, `init()`, `work()`, `add(int[], int)`, and `check()`.
- Bottom Status Bar:** Shows "Writable", "Smart Insert", and "46 : 1".

Conditional Breakpoints (cont'd)



Conditional Breakpoints (cont'd)

1. Press "Resume".

2. The program paused at the third iteration.

```
/**
 * 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
```

Terminate debugger

The screenshot shows the Eclipse IDE interface during a debug session. The top toolbar contains various icons, with a red box highlighting the 'Terminate' button (a red square with a white 'X'). The bottom toolbar also has a red box highlighting the same 'Terminate' button. A red arrow points from the top button to the bottom button, with the text "Press this or that button." written in red.

The IDE shows the following details:

- Debug Console:** Shows the execution stack with "Lab2Driver at localhost:2303" and "Thread [main] (Suspended (breakpoint at line 48 in Lab2))".
- Variables View:** Shows variables: "this" (Lab2 (id=17)), "rand" (Random (id=18)), and "i" (2).
- Source Editor:** Shows the code for "Lab2Driver.java" with the following snippet:

```
/**  
 * 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;  
    }  
}
```
- Outline View:** Shows the class structure for "Lab2" with methods: "init()", "work()", "add(int[], int)", and "check()".
- Console:** Shows the application name and path: "Lab2Driver [Java Application] D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 下午12:34:43)".

Questions?