

CS2334 Lab2

Eclipse And Debugging Survival Guide

Yu-Hsin Li and Mark Woehrer

Jan. 24, 08

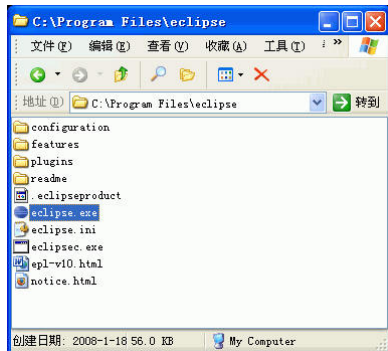
Basic Eclipse Tutorial

Basic Eclipse Debugging

Basic Eclipse Tutorial

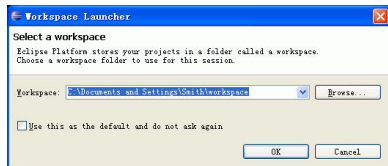
Start

- ▶ Extract the file eclipse-java-europa-fall2-win32.zip to C:\Program Files.
- ▶ Go to C:\Program Files\eclipse.
- ▶ Double click on eclipse.exe.

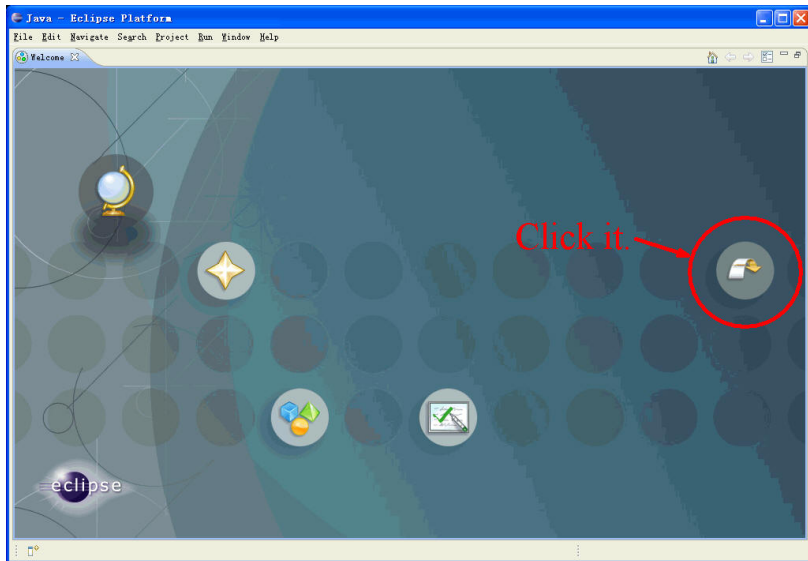


Choose a Workspace

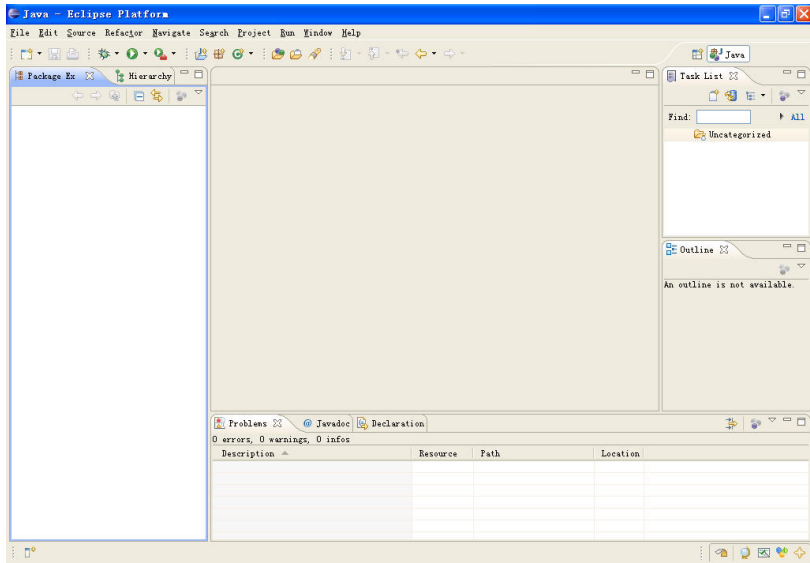
- ▶ A workspace is a folder/directory in your hard drive.
- ▶ Each workspace houses a collection of projects.
- ▶ Click OK for the default one.



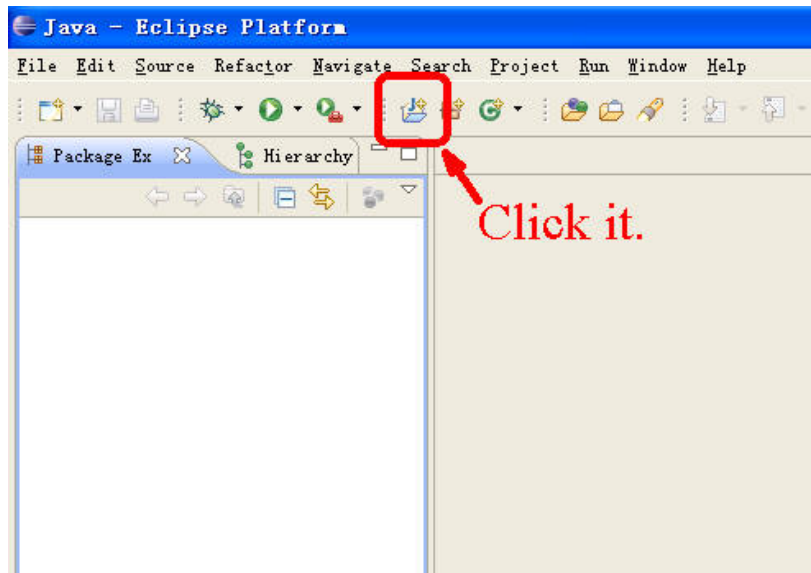
The Welcome Screen



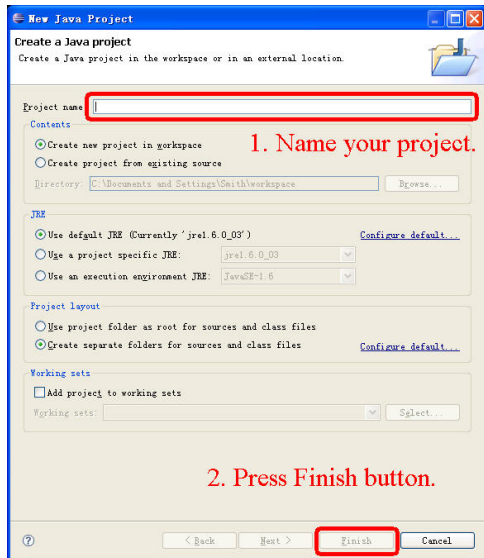
The Main Screen



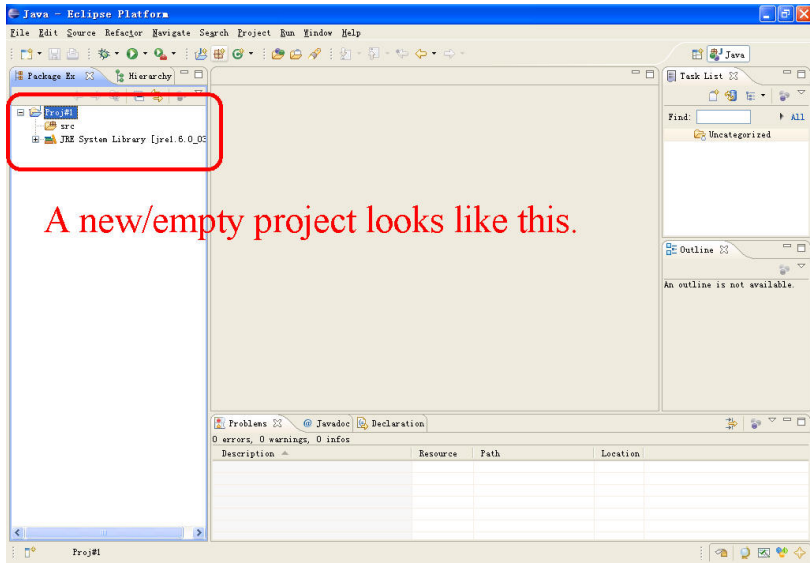
Create a New Project



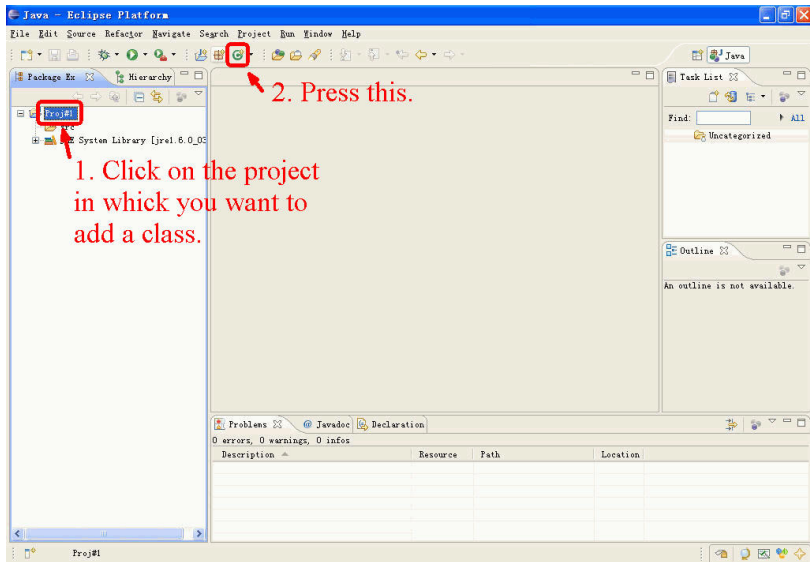
Create a New Project (Cont'd)



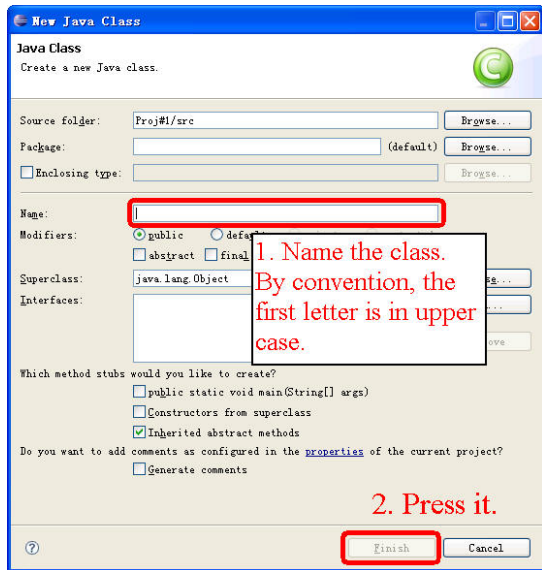
Create a New Project (Cont'd)



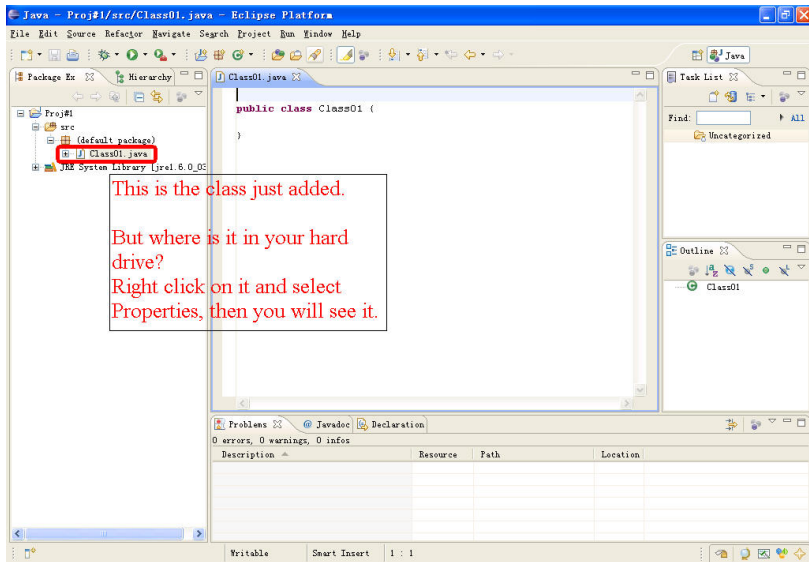
Add a Class to a Project



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 Explorer Hierarchy

Proj#1

- src
- (default package)
- Class01.java**
- JRE System Library [jre1.8.0_05]

```
public class Class01 {  
  
}
```

Task List

Find: All

Uncategorized

Outline

Class01

Problems @ Javadoc Declaration

0 errors, 0 warnings, 0 infos

Description	Resource	Path	Location

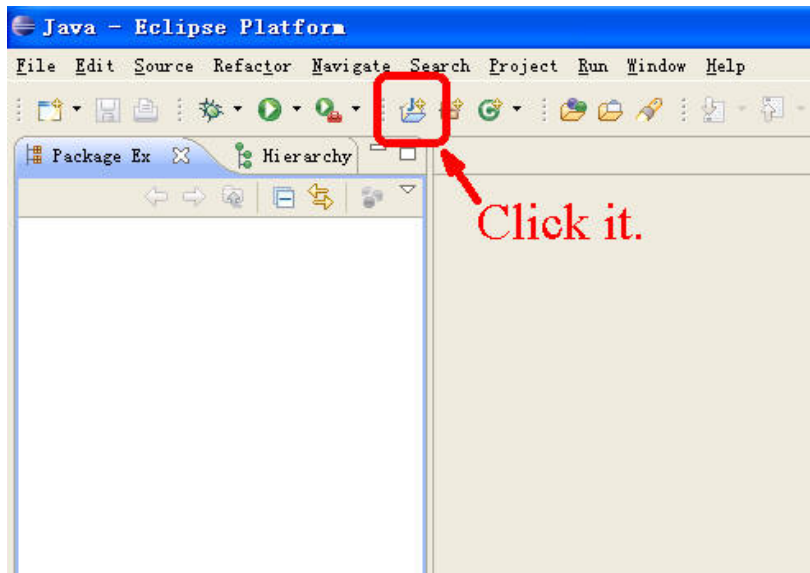
Writable Smart Insert 1 : 1

This is the class just added.

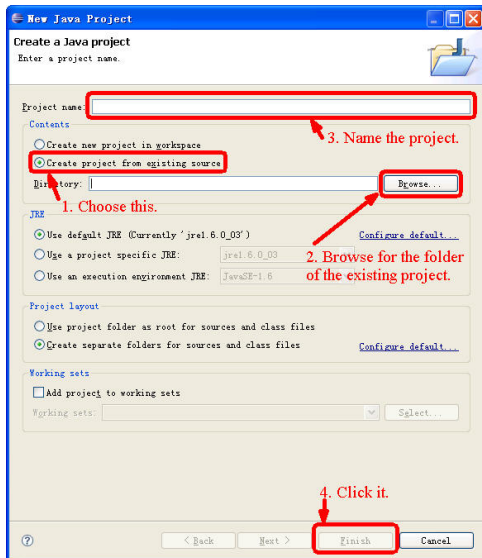
But where is it in your hard drive?

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

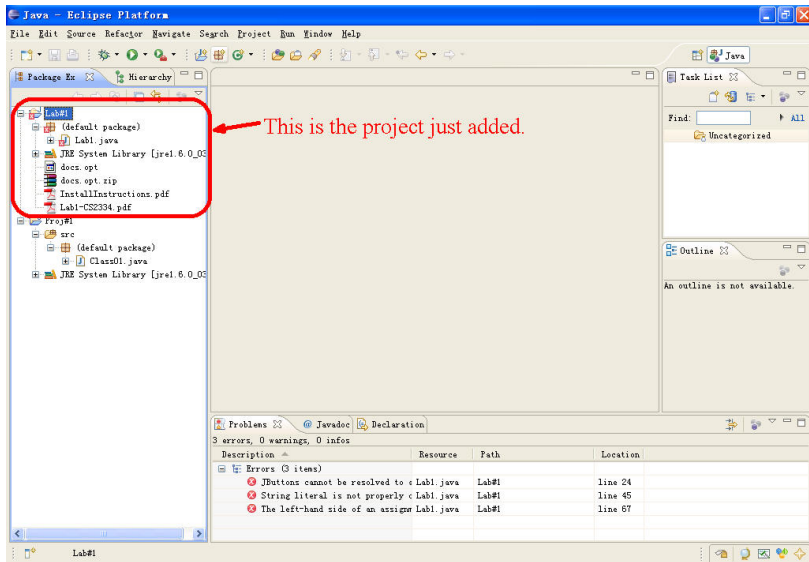
Add a Directory As a New Project



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



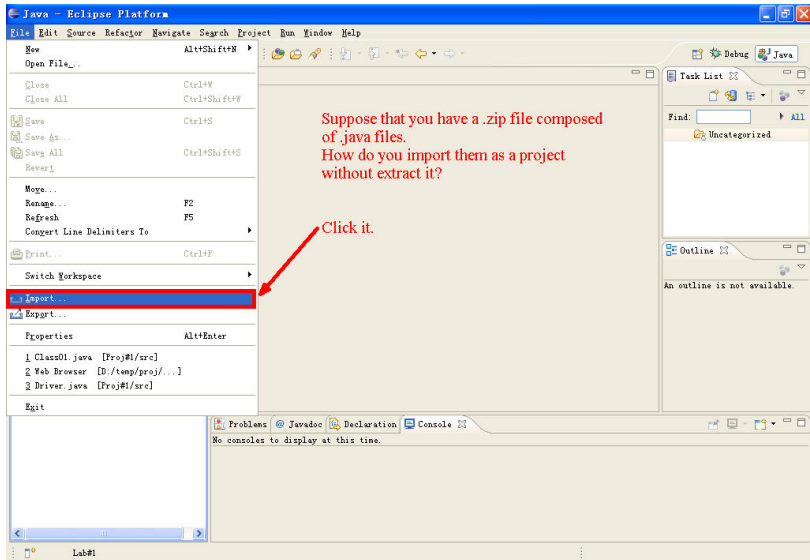
Add a Directory As a New Project (Cont'd)



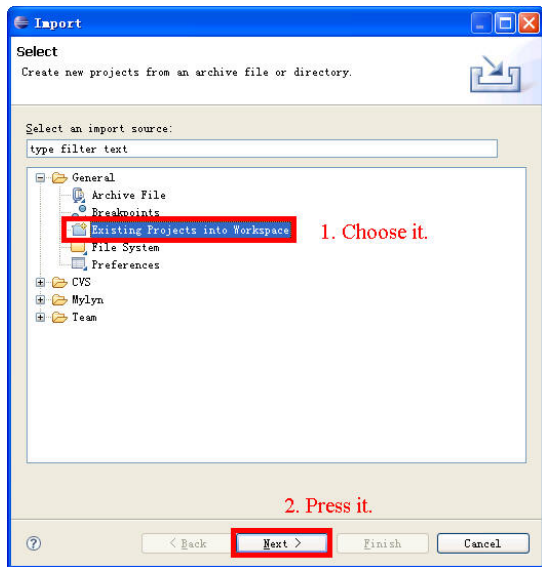
The screenshot shows the Eclipse IDE interface. The Package Explorer on the left displays a project named 'Lab#1' which is highlighted with a red box. A red arrow points to this box with the text 'This is the project just added.' The project structure includes a '(default package)' containing 'Lab1.java', and a 'JRE System Library [jre1.6.0_05]' containing 'docs.opt', 'docs.opt.zip', 'InstallInstructions.pdf', and 'Lab1-CS2334.pdf'. Below the Package Explorer, the Problems view 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

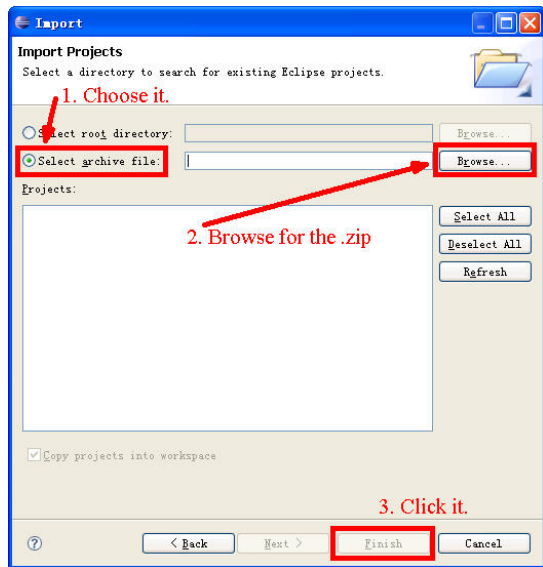
Add a .zip File As a New Project



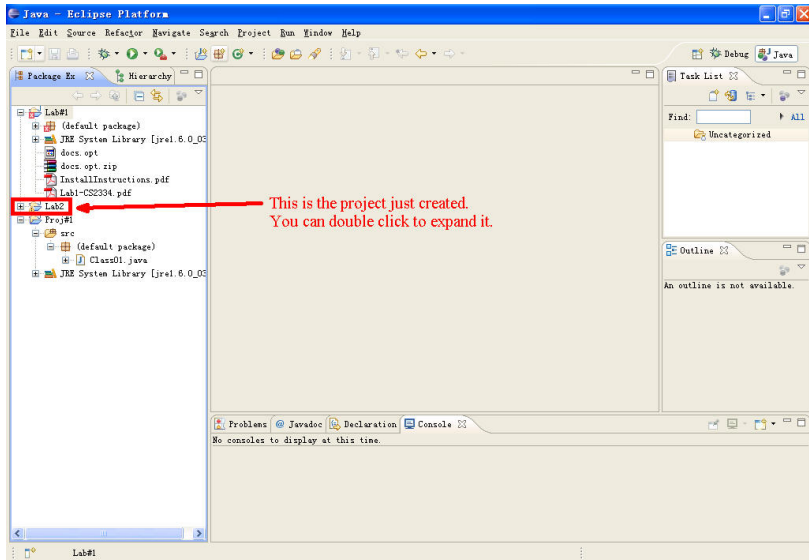
Add a .zip File As a New Project (Cont'd)



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 JButtons  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:

- A red circle highlights the error icon in the Package Explorer.
- A red circle highlights the error icon in the code editor next to the `JButtons` error.
- A red circle highlights the error icon in the code editor next to the string literal error.
- A red circle highlights the error icon in the code editor next to the assignment error.
- A red circle highlights the `Problems` view tab.
- Red arrows point from the text annotations to these elements.

Text annotations:

- "Move mouse over here, Eclipse will pop up the error message." (points to the Package Explorer error icon)
- "All errors are shown here." (points to the Problems view)
- "These are indices of errors. You can click on it." (points to the error icons in the code editor)

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 quick fix menu is displayed over the first error.

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. */
private JButton exitButton;
private JButtons jButton1;

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:

- 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 View:

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	Lab1.java	Lab#1	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 error.
- A red box highlights the quick fix menu.

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 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.
 * <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.
     */
    exitButton = new JButton( "Click Here to Exit" );
}
```

Description	Resource	Path	Location
Errors (3 items)			
JButtons cannot be resolved to a class	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 cannot be a lambda expression, an array, or a package name	Lab1.java	Lab#1	line 67

How to Run a Project

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

2. Choose this.

```

import javax.swing.*;

/**
 * Lab #1
 */

/** INSERT YOUR LAB SECTION NUMBER HERE ***
 * THE GOES HERE ***
 *
 * a sample program with a graphical user
 *
 * THE GOES HERE ***
 */

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

public class Lab1 implements ActionListener {

    // The main method of the program. */
    public static void main(String[] args) {
        JFrame windowFrame;
        windowFrame = new JFrame("Lab1");
        windowFrame.add(new JLabel("A message to be displayed in the window"));
        windowFrame.add(new JLabel("textLabel"));
        windowFrame.add(new JButton("exitButton"));
        windowFrame.add(new JButton("exitButton"));
        windowFrame.setVisible(true);
        windowFrame.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        });
        windowFrame.pack();
        windowFrame.setLocation(100, 100);
        windowFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        windowFrame.setVisible(true);
    }
}
    
```

Location	Line
Lab1.java	line 85

Run a Project with Command Line Arguments

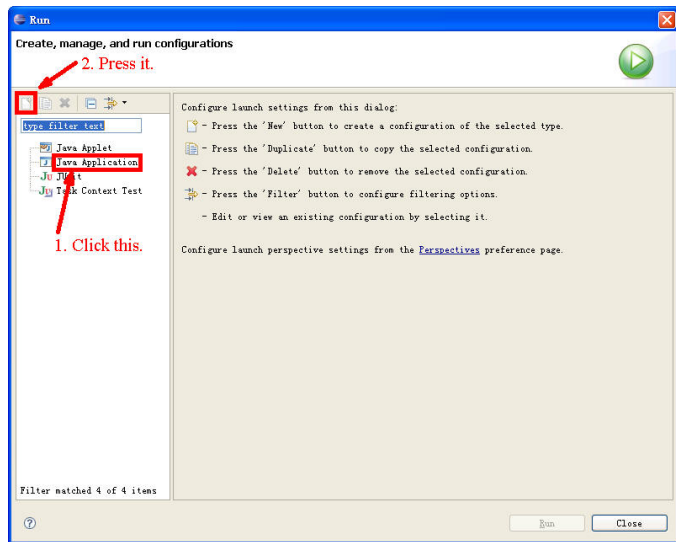
The screenshot shows the Eclipse IDE interface. In the Package Explorer on the left, a project named 'Lab#1' is expanded to show a package 'src' containing a file 'Lab1.java'. A right-click context menu is open over 'Lab1.java'. The 'Run As' option is selected, and its sub-menu is visible, with 'Open Run Dialog...' highlighted. A red arrow points from a text box to the 'Lab1.java' file, and another red arrow points from a text box to the 'Open Run Dialog...' option. The main editor window shows the source code of 'Lab1.java', which includes a 'main' method. The Outline view on the right shows the class structure, including 'main(String[])'. The Run As dialog table at the bottom shows a 'Java Application' configuration with the location 'line 85'.

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

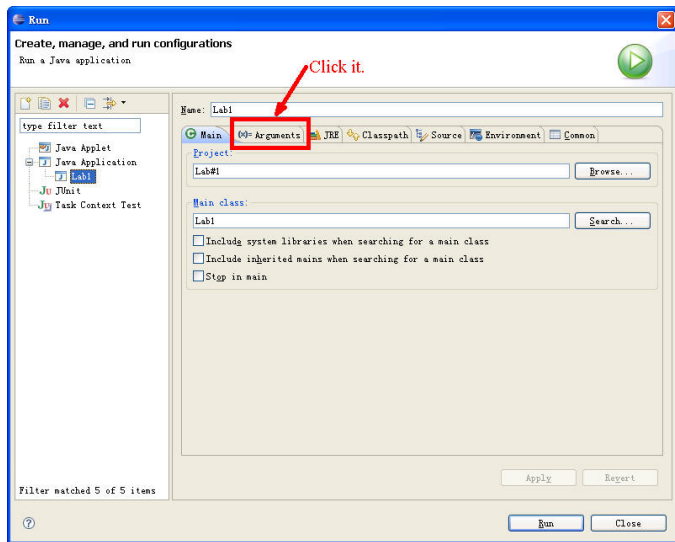
2. Click this.

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)



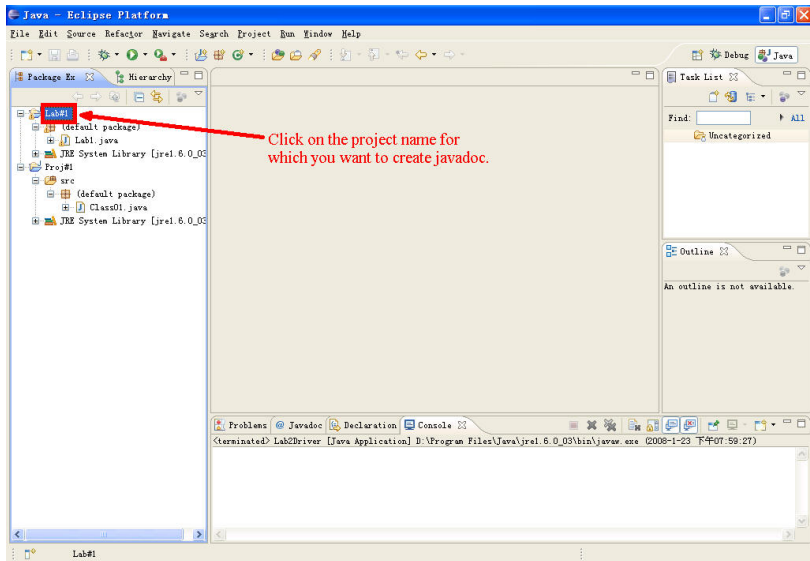
Run a Project with Command Line Arguments (cont'd)

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" configuration is selected, and the "Arguments" tab is active. The "Name:" field contains "Lab1". The "Program arguments:" field is highlighted with a red rectangle and contains the text "input1.txt input2.txt". A red arrow points from the text "1. Place all arguments here." to this field. Below it, the "VM arguments:" field is empty. The "Working directory:" section has "Default:" selected with the value "\${workspace_loc:Lab1#1}". At the bottom right, the "Run" button is highlighted with a red rectangle, and a red arrow points from the text "2. Click it." to it. Other buttons include "Apply", "Revert", and "Close".

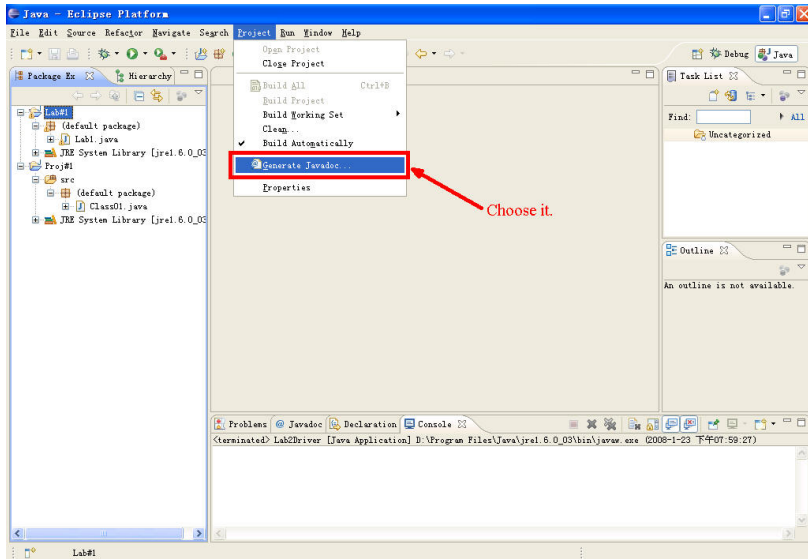
1. Place all arguments here.

2. Click it.

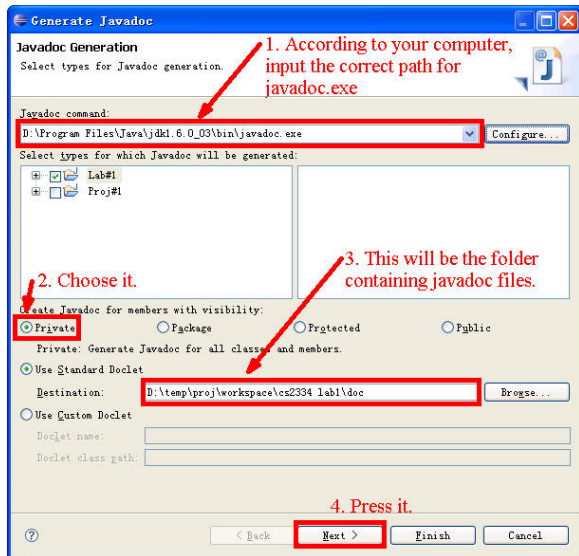
Create Javadoc



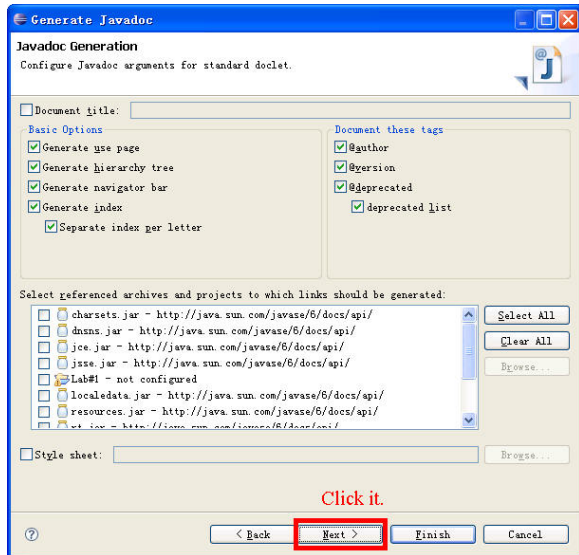
Create Javadoc (cont'd)



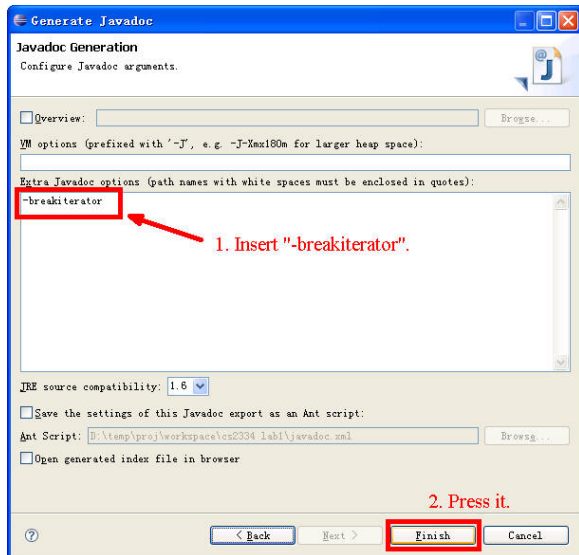
Create Javadoc (cont'd)



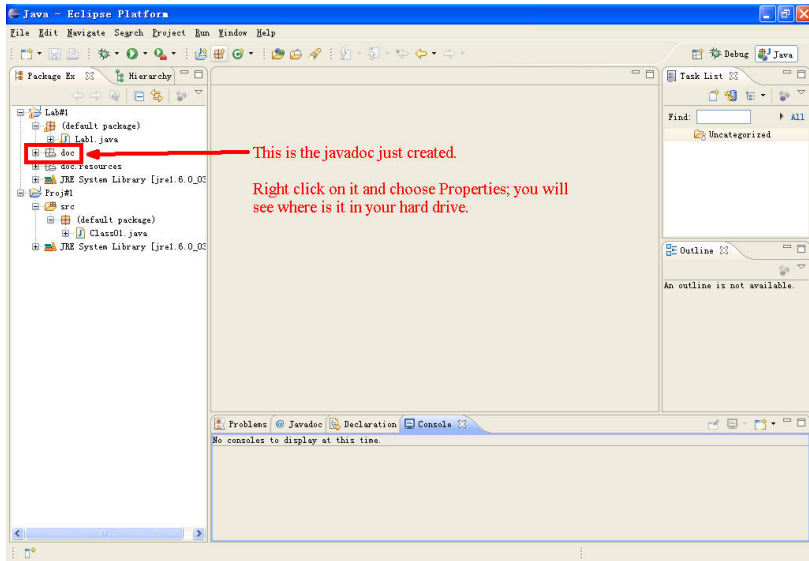
Create Javadoc (cont'd)



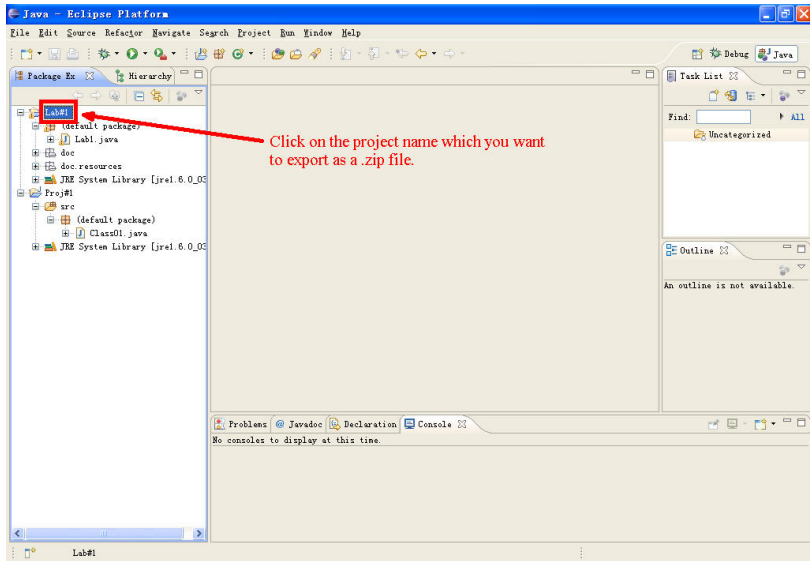
Create Javadoc (cont'd)



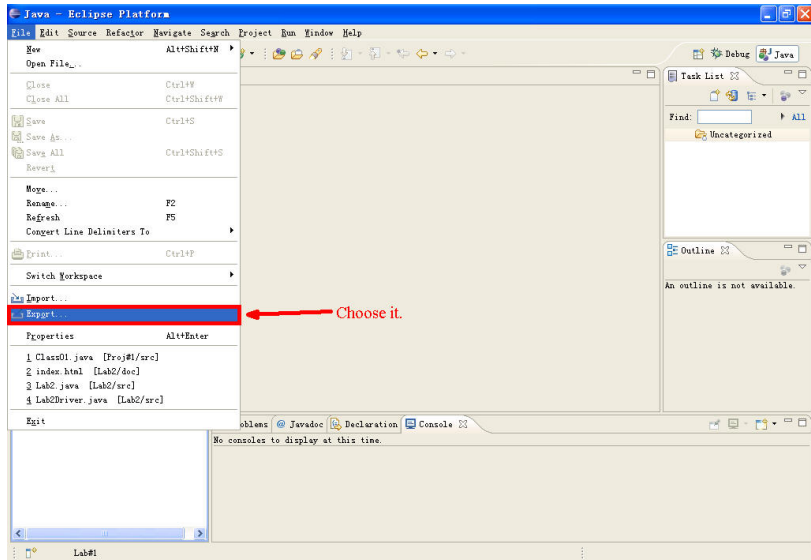
Create Javadoc (cont'd)



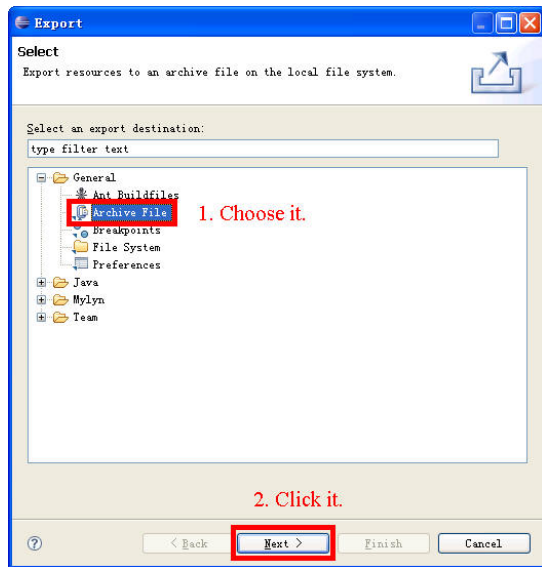
Export to a .zip File



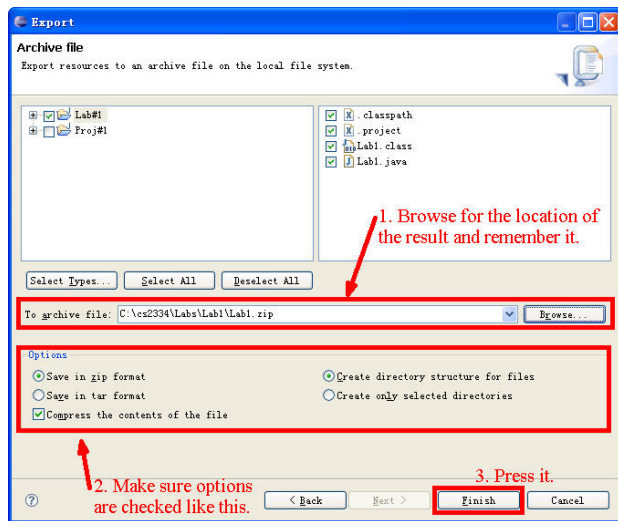
Export to a .zip File (cont'd)



Export to a .zip File (cont'd)

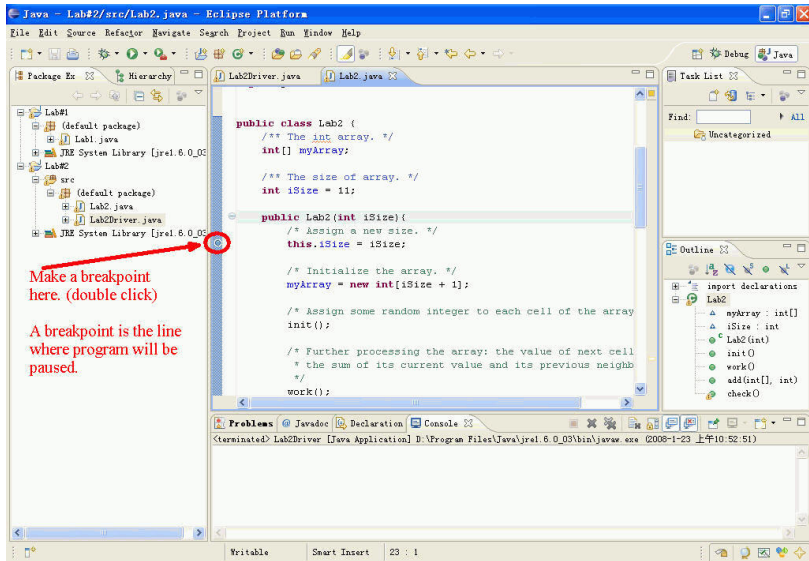


Export to a .zip File (cont'd)



Basic Eclipse Debugging

Breakpoints



The screenshot shows the Eclipse IDE with the following components:

- Package Explorer:** Shows a project structure with 'Lab#2' containing a 'src' package with 'Lab2.java' and 'Lab2Driver.java'. A red circle highlights a breakpoint icon on the 'Lab2.java' file.
- Editor:** Displays the source code of 'Lab2.java'. The constructor method is highlighted in light blue. The code is:

```
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:** Shows 'Uncategorized'.
- Outline:** Shows the class structure with methods like 'init()', 'work()', 'add(int[], int)', and 'check()'. The constructor 'Lab2(int)' is highlighted with a green circle.
- Console:** Shows a 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 image:

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

Run Eclipse Debugger

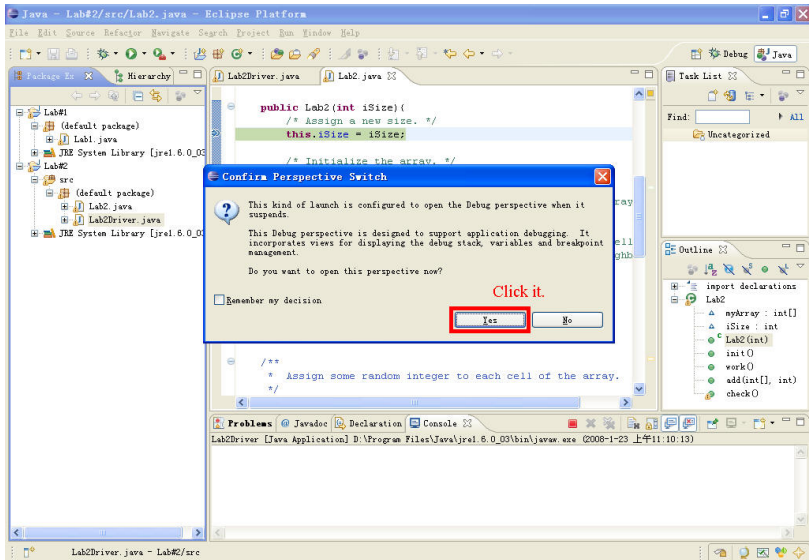
The screenshot shows the Eclipse IDE interface with the following elements:

- Window Title Bar:** Java - Lab#2/src/Lab2.java - Eclipse Platform
- Menu Bar:** File Edit Source Refactor Navigate Search Project Run Window Help
- Package Explorer:** Shows a project structure with Lab#2 containing a src folder with Lab2.java and Lab2Driver.java. Lab2Driver.java is highlighted with a red box.
- Context Menu:** Opened over Lab2Driver.java. The 'Run As' option is expanded, and '1. Java Application' is highlighted with a red box.
- Code Editor:** Displays the source code of Lab2.java, including a main method that iterates through an array.
- Task List:** Shows 'Uncategorized'.
- Outline:** Shows the class structure of Lab2, including methods like init(), work(), add(), and check().
- Console:** Shows the output of the Java application, including the path to the Java runtime.

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

2. Click it.

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
```

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
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

```

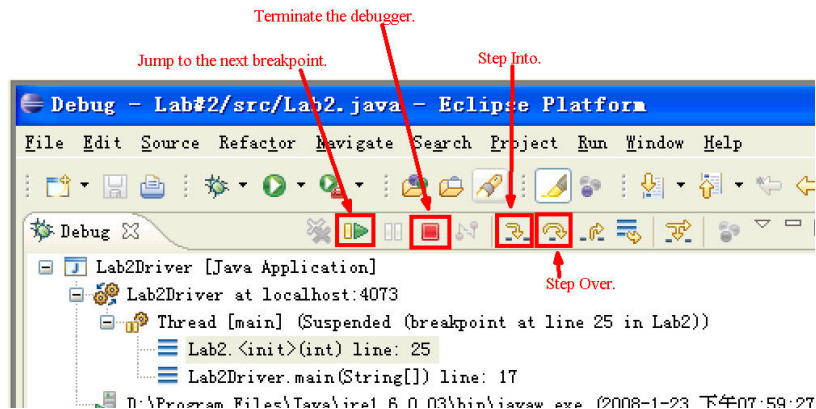
Console [Java Application] D:\Program Files\Java\jre1.6.0_03\bin\javaw.exe (2008-1-23 上午11:10:13)

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.

Writable Smart Insert 25 : 1

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.

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

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)

00= Variables

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

Breakpoints

Lab2Driver.java Lab2.java

```
/* 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();
}
```

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11:29:48)

Writable Smart Insert 31 : 1

Step Into (cont'd)

The screenshot shows the Eclipse IDE in a debug state. The top toolbar includes buttons for 'Debug' and 'Java'. The 'Debug Console' on the left shows the execution flow:

- Lab2Driver [Java Application]
- Lab2Driver at localhost:2893
- Thread [main] (Suspended)
- Lab2.init() line: 46
- Lab2.<init>(int) line: 31
- Lab2Driver.main(String[]) line: 17

The 'Variables' view on the right shows a table with the following content:

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;
    }
}
```

The 'Outline' view on the right shows the class structure:

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

The 'Console' view at the bottom shows the output of the application:

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

A red box highlights the `init()` method in the code editor, and a red arrow points to it with the text "We step into the init() method."

Conditional Breakpoints

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

File Edit Source Refactor Navigate Search Project Run Window Help

Debug Variables Breakpoints

00= Variables Value

Name	Value
this	Lab2 (id=17)

Lab2Driver [Java Application]

- Lab2Driver at localhost:1421
 - Thread [main] (Suspended)
 - Lab2.init() line: 46
 - Lab2.<init>(int) line: 31
 - Lab2Driver.main(String[]) line: 17

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
```

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

First, make a breakpoint.

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12:16:25)

Writable Smart Insert 46 : 1

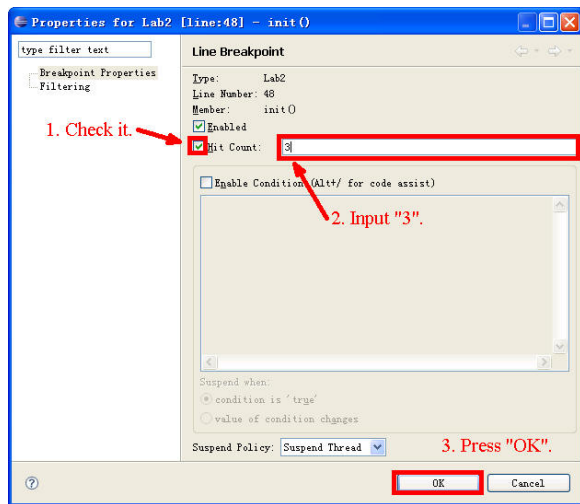
Conditional Breakpoints (cont'd)

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 `Breakpoint Properties...` option is highlighted with a red box. A red arrow points to this option with the text "Right click on the breakpoint, then select this." The IDE interface includes a Debug console, a Variables view showing `this` with value `Lab2 (id=17)`, and an Outline view showing the class structure.

```
/**
 * 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;
    }
}
```

Breakpoint Properties...

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 contains the same 'Terminate' button, also highlighted with a red box. A red arrow points from the top button to the bottom button, with the text "Press this or that button." next to it.

The IDE shows the following components:

- Debug Console:** Shows the execution stack with "Lab2Driver [Java Application]" at the top, followed by "Thread [main] (Suspended (breakpoint at line 48 in Lab2))", "Lab2.init() line: 48", "Lab2.<init>(int) line: 31", and "Lab2Driver.main(String[]) line: 17".
- Variables View:** Shows variables: "this" (Lab2 (id=17)), "rand" (Random (id=18)), and "i" (2).
- Source Editor:** Shows the code for "Lab2Driver.java" and "Lab2.java". The code in "Lab2.java" is:

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

Questions?