SUPPLEMENT – ECLIPSE TUTORIAL (2)

It is normal to make many mistakes as you write programs. Debugging is a cyclic process of editing, compiling, and fixing errors.

> "As soon as we started programming, we found out to our surprise that it wasn't as easy to get programs right as we had thought. I can remember the exact instant when I realized that a large part of my life from then on was going to be spent in finding mistakes in my own programs."
>
> - Maurice Wilkes

This tutorial will help you to become familiar with debugger utility of Eclipse, including setting breakpoints and watching the values stored in variables.

1. Setting Breakpoints

The debugger executes every line until it encounters a *breakpoint*, and the execution of your program will pause at a breakpoint. You can trace the part of the program at the breakpoint. Using the breakpoint, you can skip over the sections work correctly and only focus on the sections may cause problems.

Set a breakpoint by choosing *Run* ➔ *Toggle Line Breakpoint* (Figure 1).

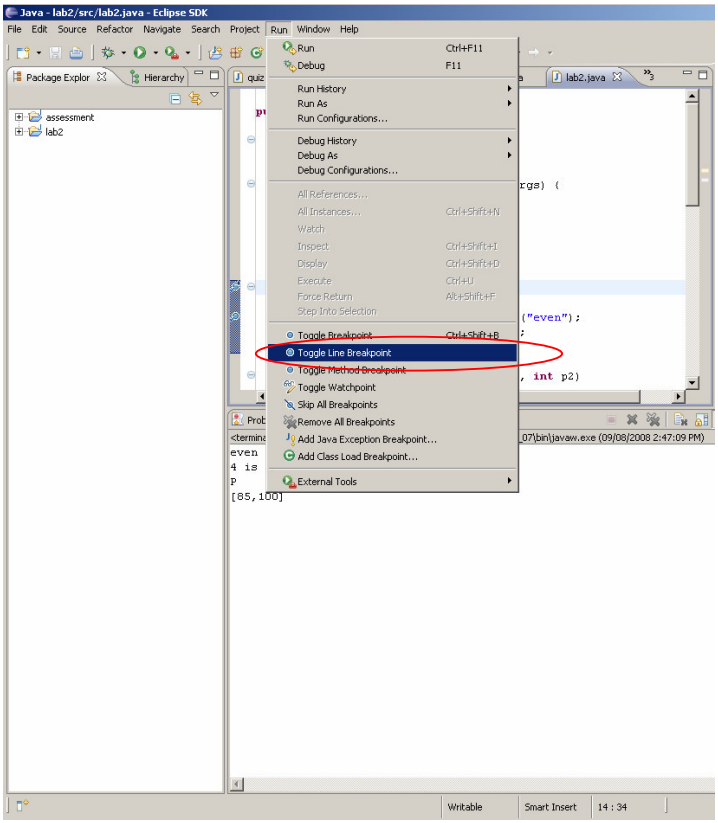


Figure 1: set a breakpoint

Remove a breakpoint by double-clicking the cutter of the line (Figure 2).

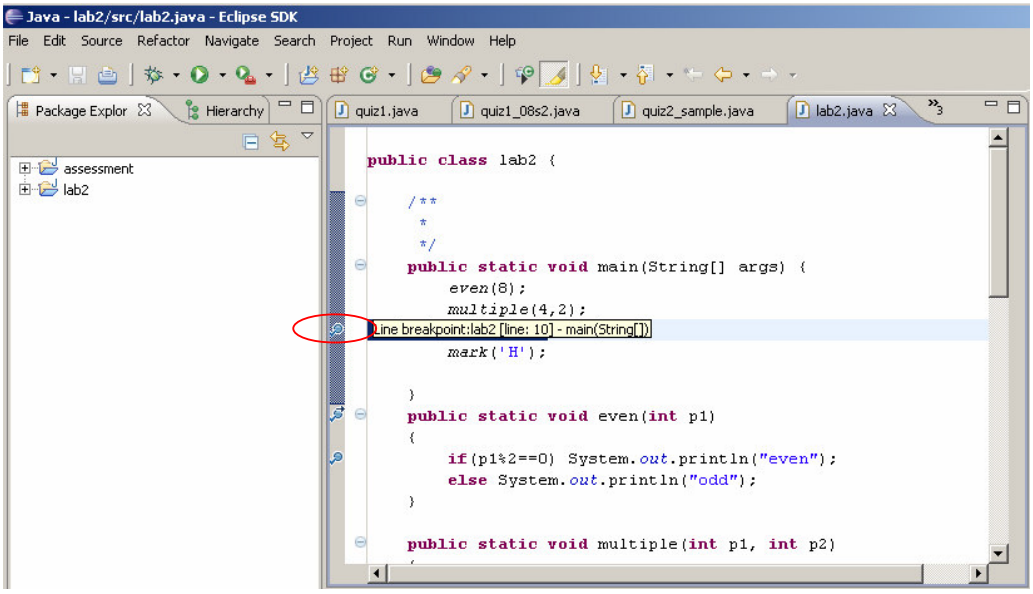


Figure 2: Remove a breakpoint

2. Starting the Debugger

i. Select the class name to be debugged in the *Package Explor*

ii. Right-click the class name ➔ choose *Debug As* ➔ *Java Application* to start debugging (Figure 3)

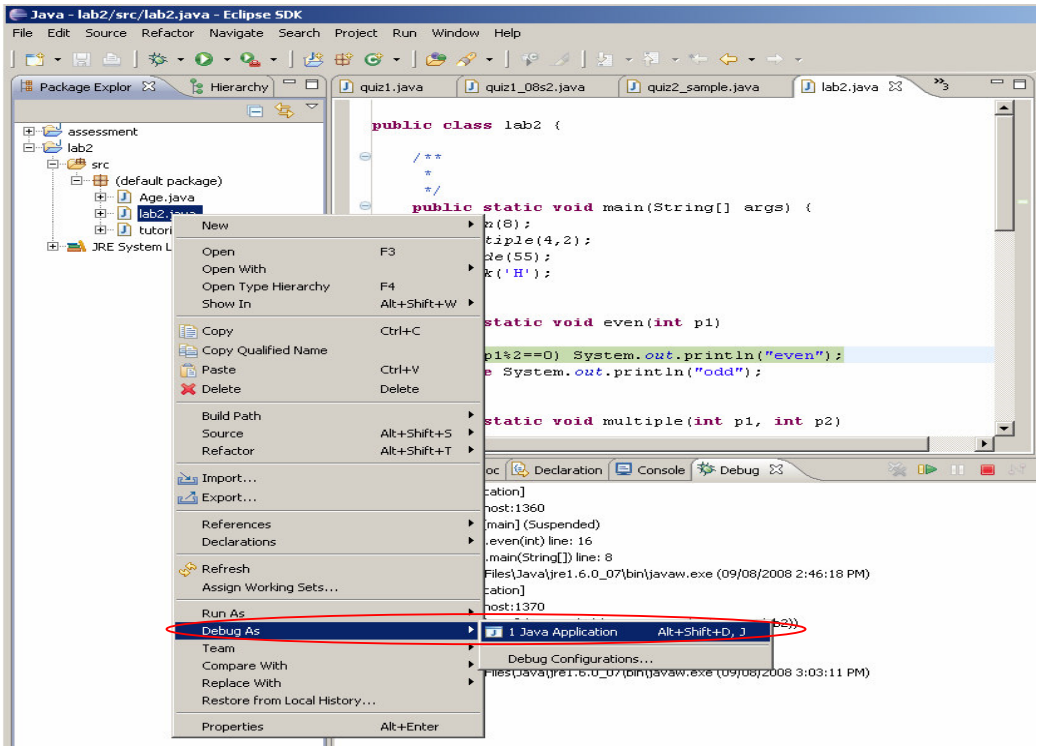


Figure 3: Start the debugger

iii. Click *Yes* button on the *Confirm Perspective Switch* dialog to switch to the Debug perspective

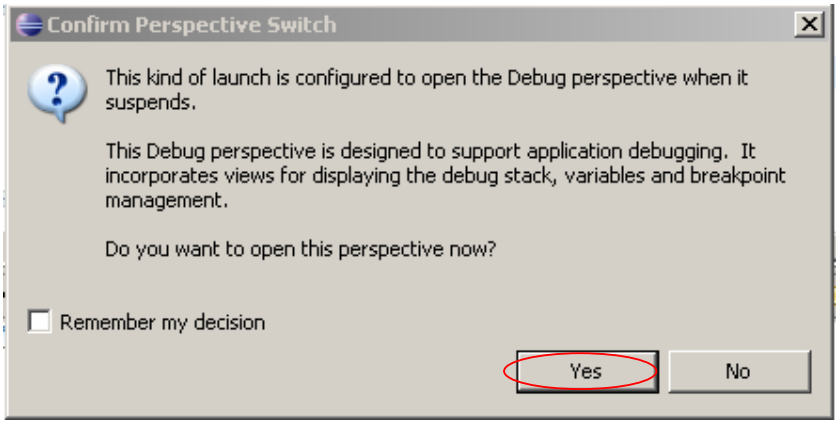


Figure 4

iv. The UI for Debug perspective is shown as below:

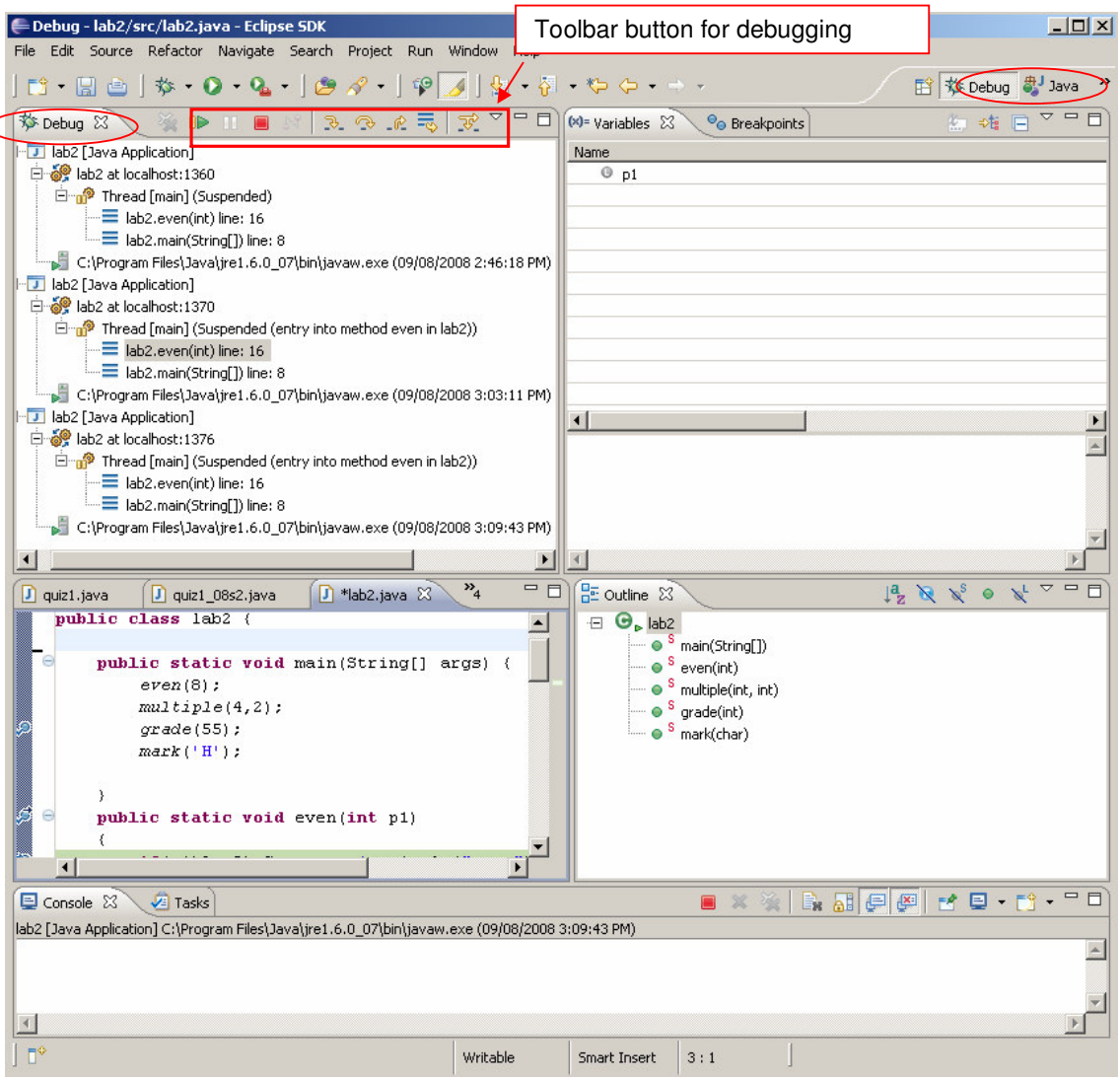


Figure 5 Debug perspective UI

3. Controlling Program Execution

When your program in debugging mode, the toolbar buttons for debugging are displayed in the Debug window (Figure 5). The debugging commands are also available in the Run menu (Figure 6). The command for controlling program execution include:

- Resume: to resume the execution of a paused program
- Suspend: to temporarily stop execution of a program
- Terminate: to end the current debugging session
- Step Into: to execute a single statement or steps into a method
- Step Over: to execute a single statement. If the statement contains a call to a method, the entire method is executed without stepping through it
- Step Return: to execute all the statements in the current method and returns to its caller
- Run to Line: to run the program, starting from the current execution point, and pauses and places the execution point of code containing the cursor, or at a breakpoint.

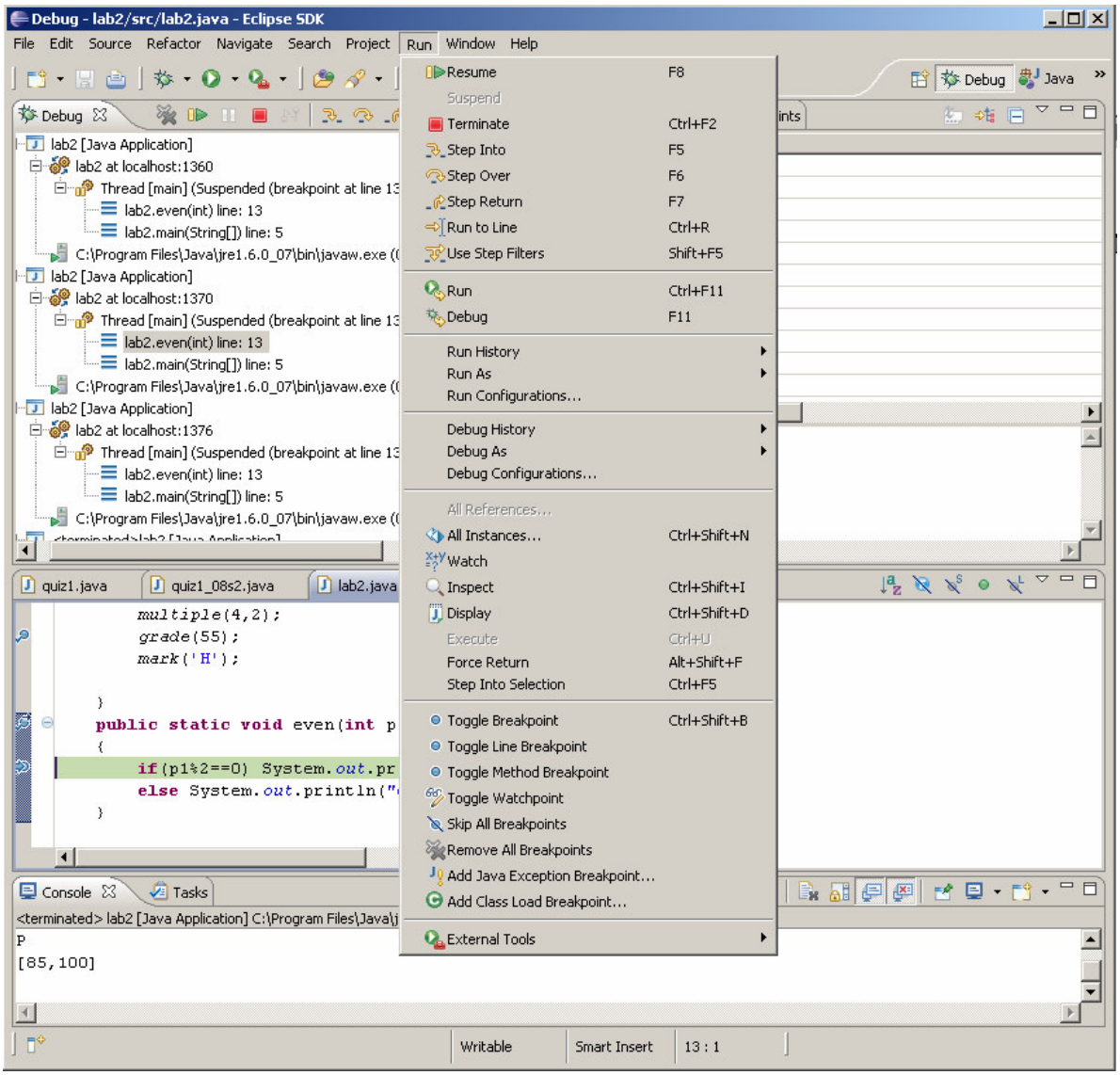
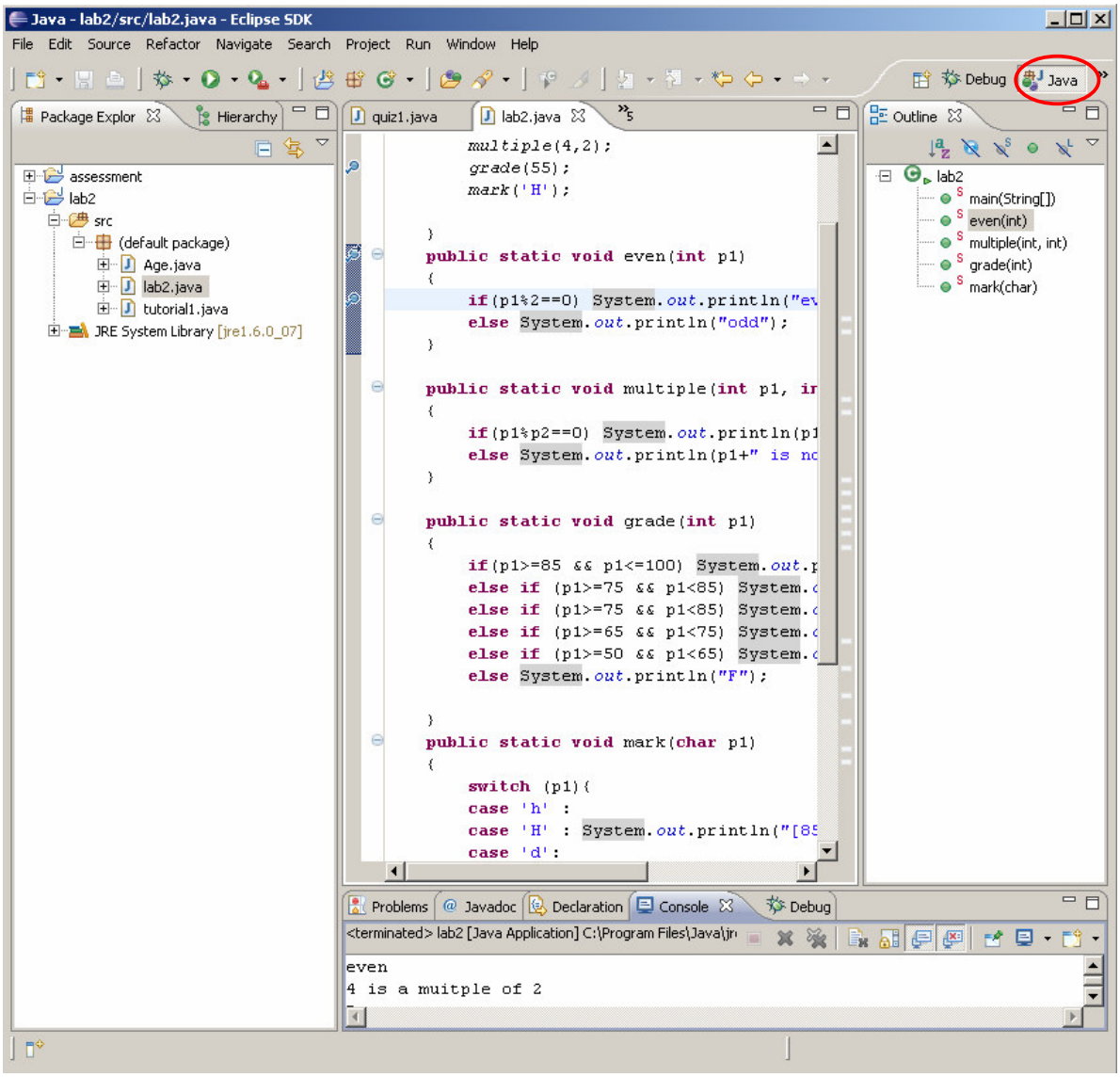


Figure 6: Debugging Commands

4.


Java - lab2/src/lab2.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Debug
Java
Package Explor
Hierarchy
assessment
lab2
src
(default package)
Age.java
lab2.java
tutorial1.java
JRE System Library [jre1.6.0_07]
quiz1.java
lab2.java
multiple(4,2);
grade(55);
mark('H');
public static void even(int p1)
if(p1%2==0) System.out.println("ev
else System.out.println("odd");
public static void multiple(int p1, ir
if(p1%p2==0) System.out.println(p1
else System.out.println(p1+" is nc
public static void grade(int p1)
if(p1>=85 && p1<=100) System.out.p
else if (p1>=75 && p1<85) System.
else if (p1>=75 && p1<85) System.
else if (p1>=65 && p1<75) System.
else if (p1>=50 && p1<65) System.
else System.out.println("F");
public static void mark(char p1)
switch (p1){
case 'h' :
case 'H' : System.out.println("[85
case 'd':
Outline
lab2
main(String[])
even(int)
multiple(int, int)
grade(int)
mark(char)
Problems
Javadoc
Declaration
Console
Debug
<terminated> lab2 [Java Application] C:\Program Files\Java\jr
even
4 is a muitple of 2