

## **Supplement II.E: Teaching Java Effectively with NetBeans**

### **For Introduction to Java Programming**

**By Y. Daniel Liang**

## **0 Introduction**

Supplement II.D, "NetBeans Tutorial," gives a brief tutorial on how to use NetBeans. NetBeans is not only a powerful Java program development tool, but it is also a valuable pedagogical tool for teaching and learning Java programming. This supplement shows how to use NetBeans effectively with the text.

The supplement is written for instructors, but it is also useful to students.

## **1 Important Tips**

The objective of the course is to teach Java, not NetBeans. NetBeans is a complex and powerful tool. All you need for this course, however, is a minimum set of features that enable students to create, compile, run, and debug programs. So students should avoid exploring unnecessary features.

If your students follow the instructions in Supplement II.D, "NetBeans Tutorial," or the instructions from you, students can master all essential skills in thirty minutes. It is important that your students adhere to the instructions to avoid frustrating mistakes. If a mistake is made, simply read the instructions and restart from scratch.

## **2 NetBeans as a Valuable Pedagogical Tool**

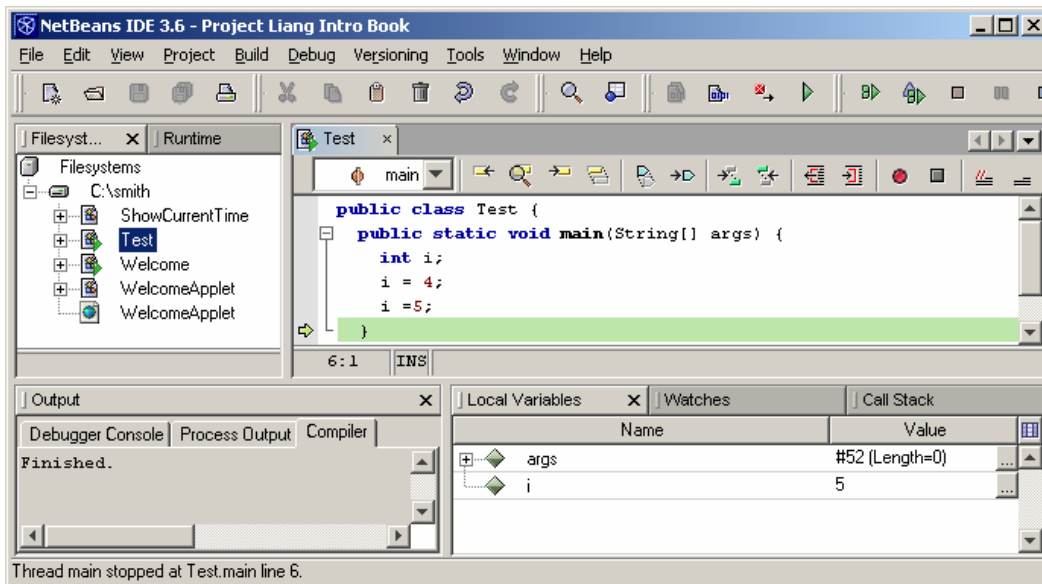
The following sections demonstrate how to utilize NetBeans in the first seven chapters.

### *2.1 Using NetBeans in Chapter 1*

After Listing 1.1, you can start to cover how to create, compile, and run a program in NetBeans. You may also introduce how to use NetBeans online help.

### *2.2 Using NetBeans in Chapter 2*

You may start to introduce debugging when you cover variables. You can use debug to show the value of a variable in the memory and show the change of the value during execution. Figure 1 shows a simple test program with variable *i*.

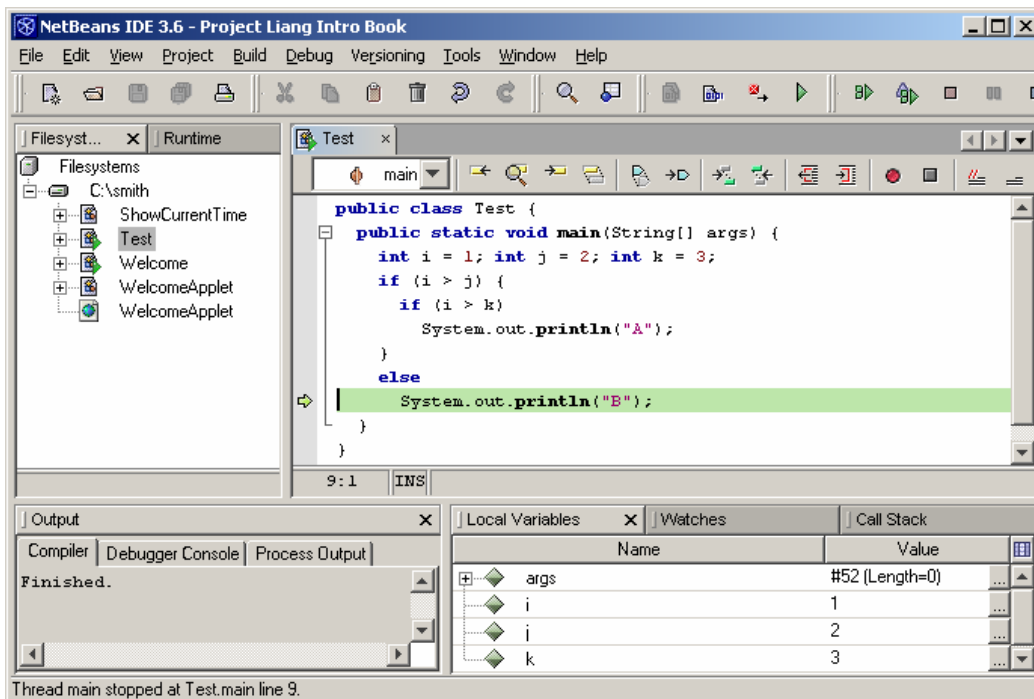


**Figure 1**

*Displaying values of variables in NetBeans debugger.*

### 2.3 Using NetBeans in Chapter 3

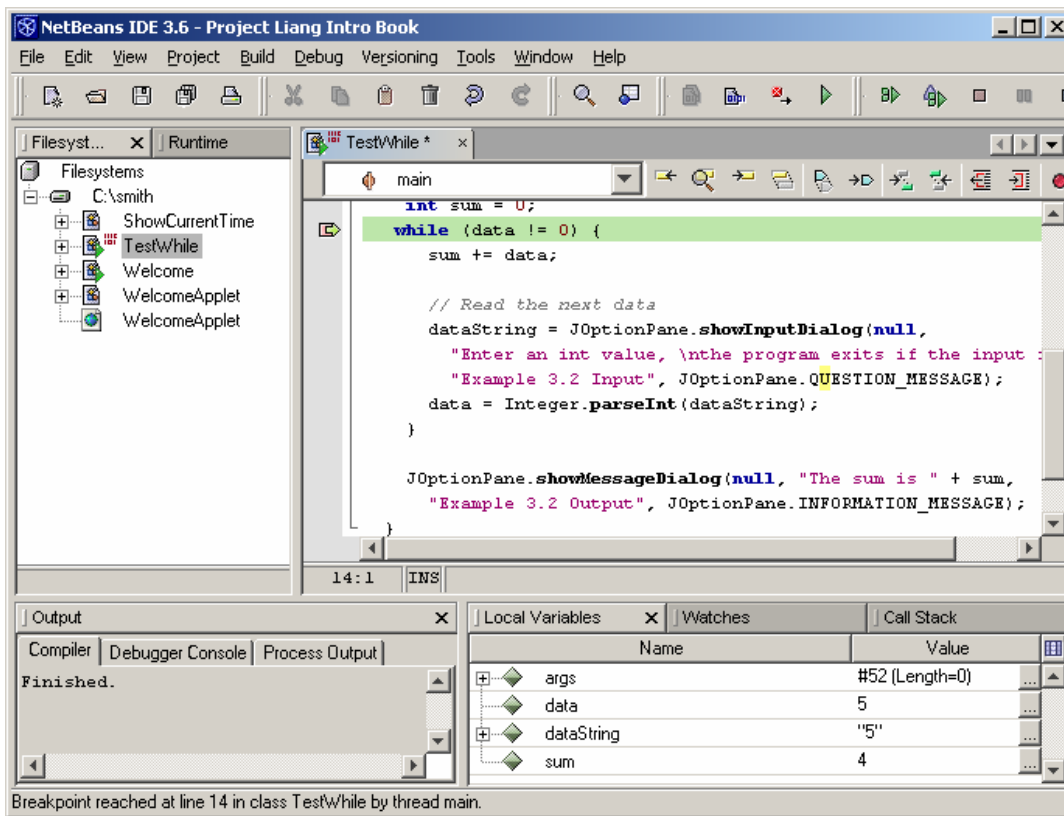
Use the debugger to trace the if statements in Section 3.2.3, "Nested if Statements," as shown in Figure 2.



**Figure 2**

*Trace the execution of an if statement.*

Use the debugger to trace the while loop in Listing 3.2 TestWhile.java (Using while Loop),” as shown in Figure 3.

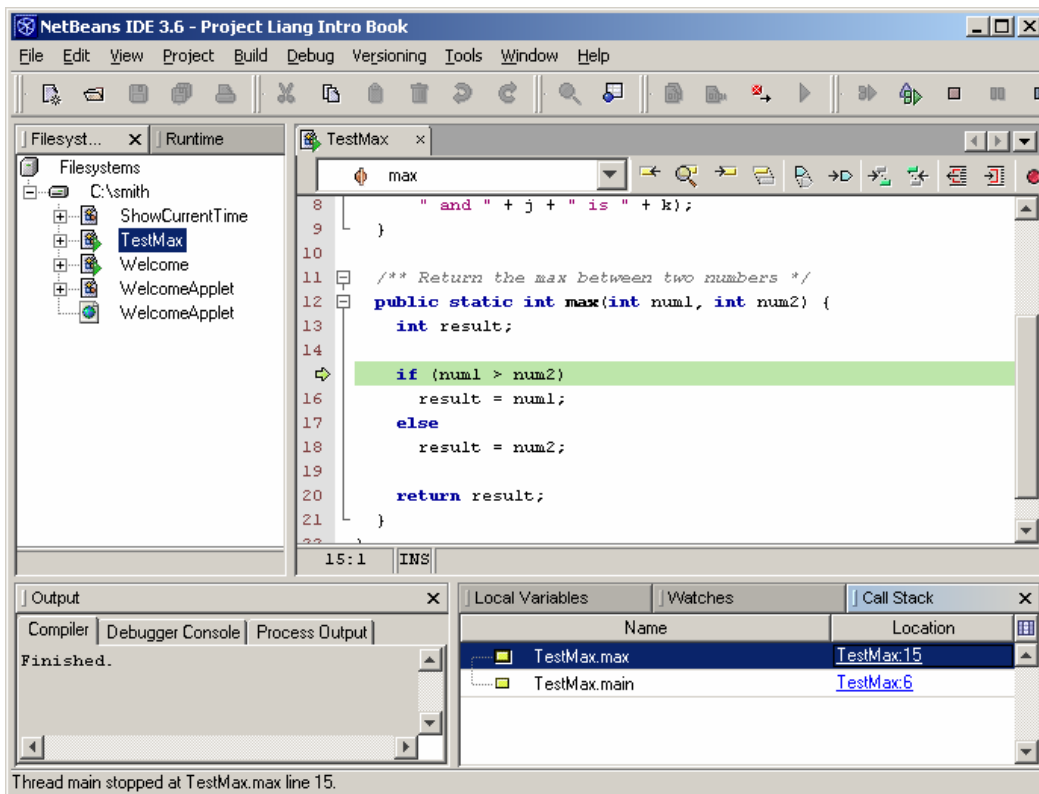


**Figure 3**

*Trace the execution of a loop statement.*

#### 2.4 Using NetBeans in Chapter 4

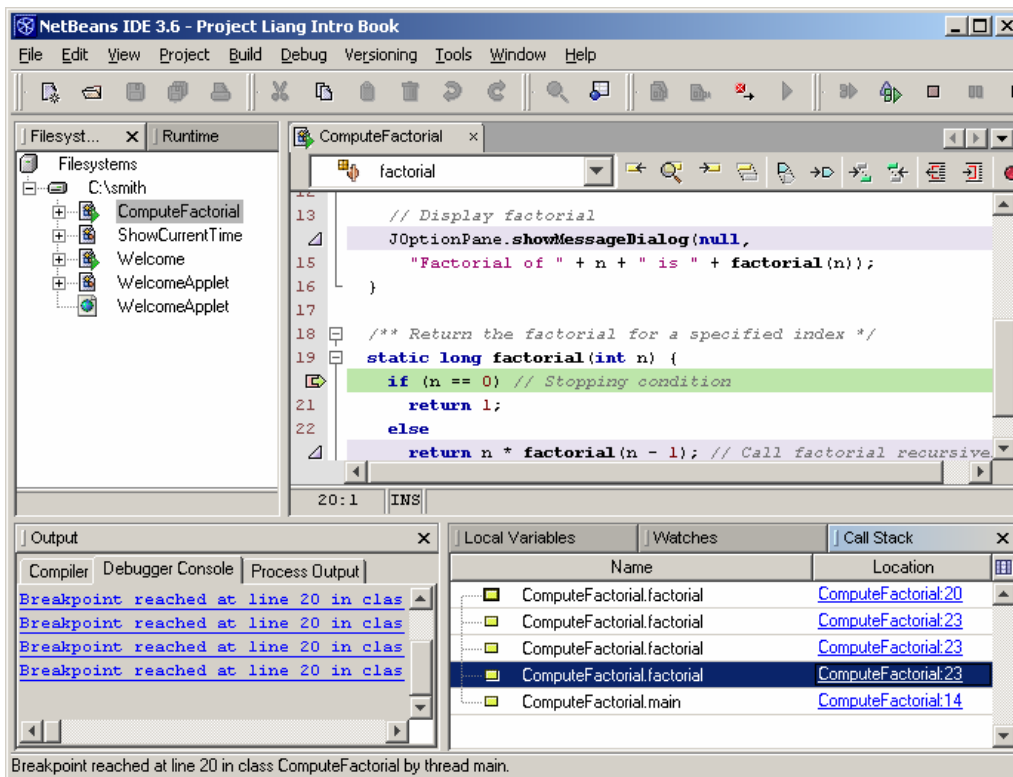
You can use the debugger to show the call stack, which is very effective to help understand method invocation. Let us use Listing 5.1 to demonstrate method invocation. Set a breakpoint at Line 6. Start debugger, and the debugger pauses at Line 6. Choose Step into to step into the max method, as shown in Figure 4. Now in the Call Stack tab of the Debugger window, you will see method max to be invoked.



**Figure 4**

*Trace method invocation.*

Figure 5 shows tracing recursive execution of the factorial method.

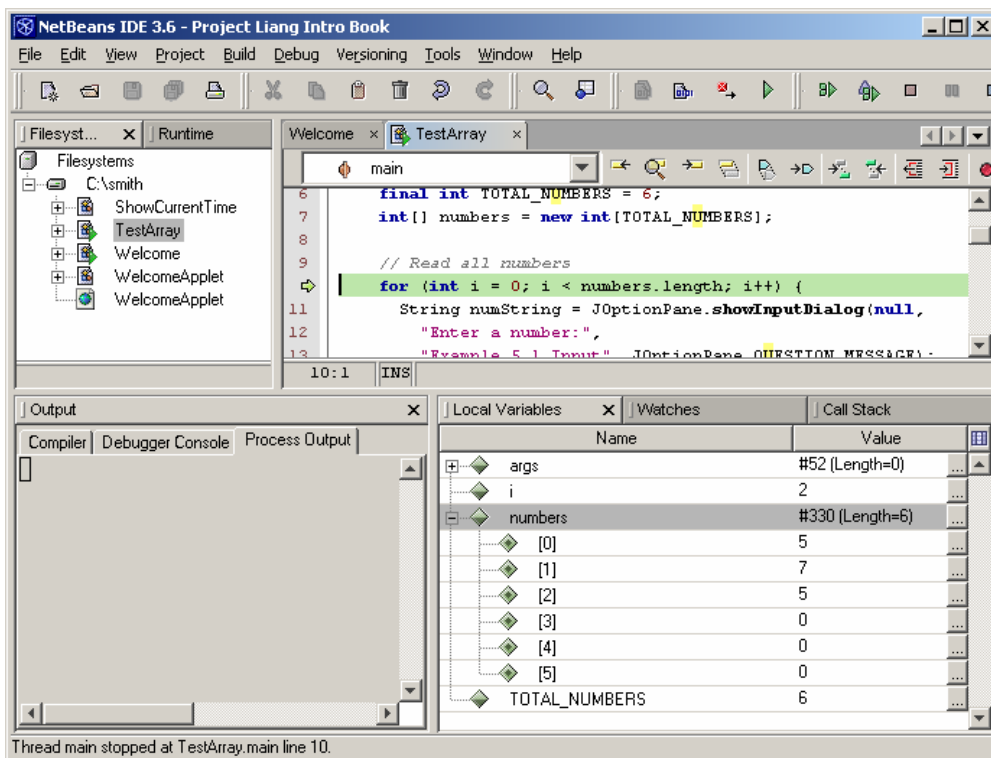


**Figure 5**

*Trace a recursive method invocation.*

### 2.5 Using NetBeans in Chapter 5

You can use the debugger to show the values of all the elements in an array. Figure 6 shows debugging Listing 6.1.



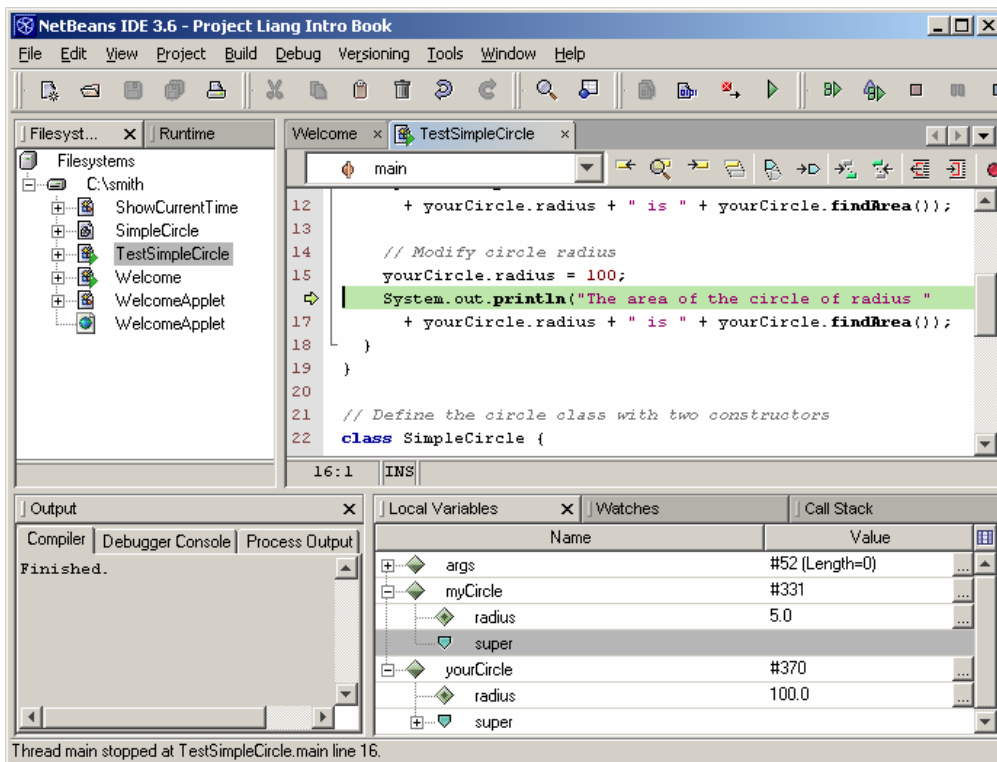
**Figure 6**

*You can see the change of values in an array.*

You can use the debugger to demonstrate how arguments are passed and to see the differences between passing primitive type values and arrays.

## 2.5 Using NetBeans in Chapter 6

You can use the debugger to show the contents of an object. Figure 7 shows debugging Listing 7.1.



**Figure 7**

*You can see the change of values in an object.*

You can use the debugger to demonstrate how arguments are passed and to see the differences between passing primitive type values and objects.