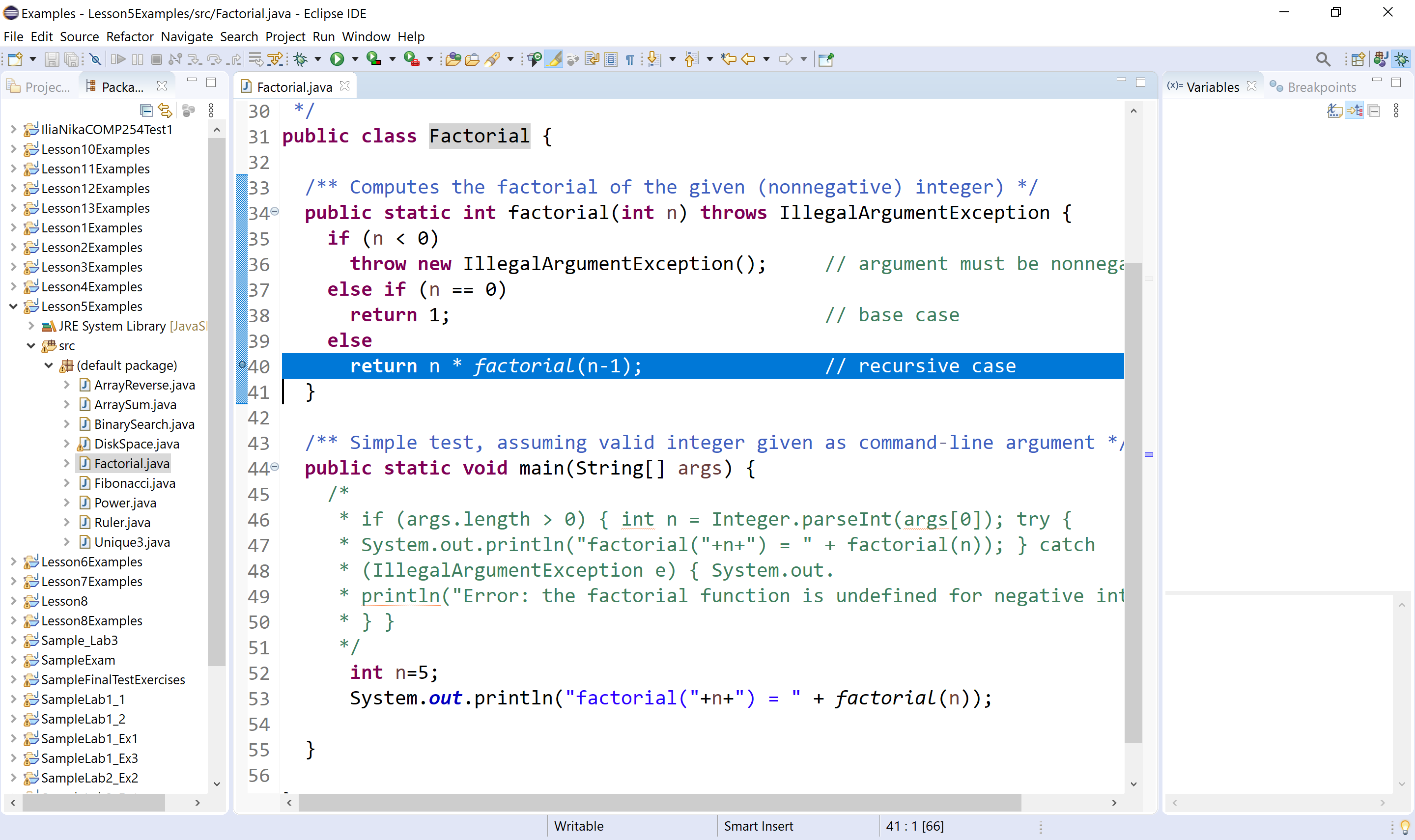
This document provides the necessary instructions for completing the Week 5 lab exercises.

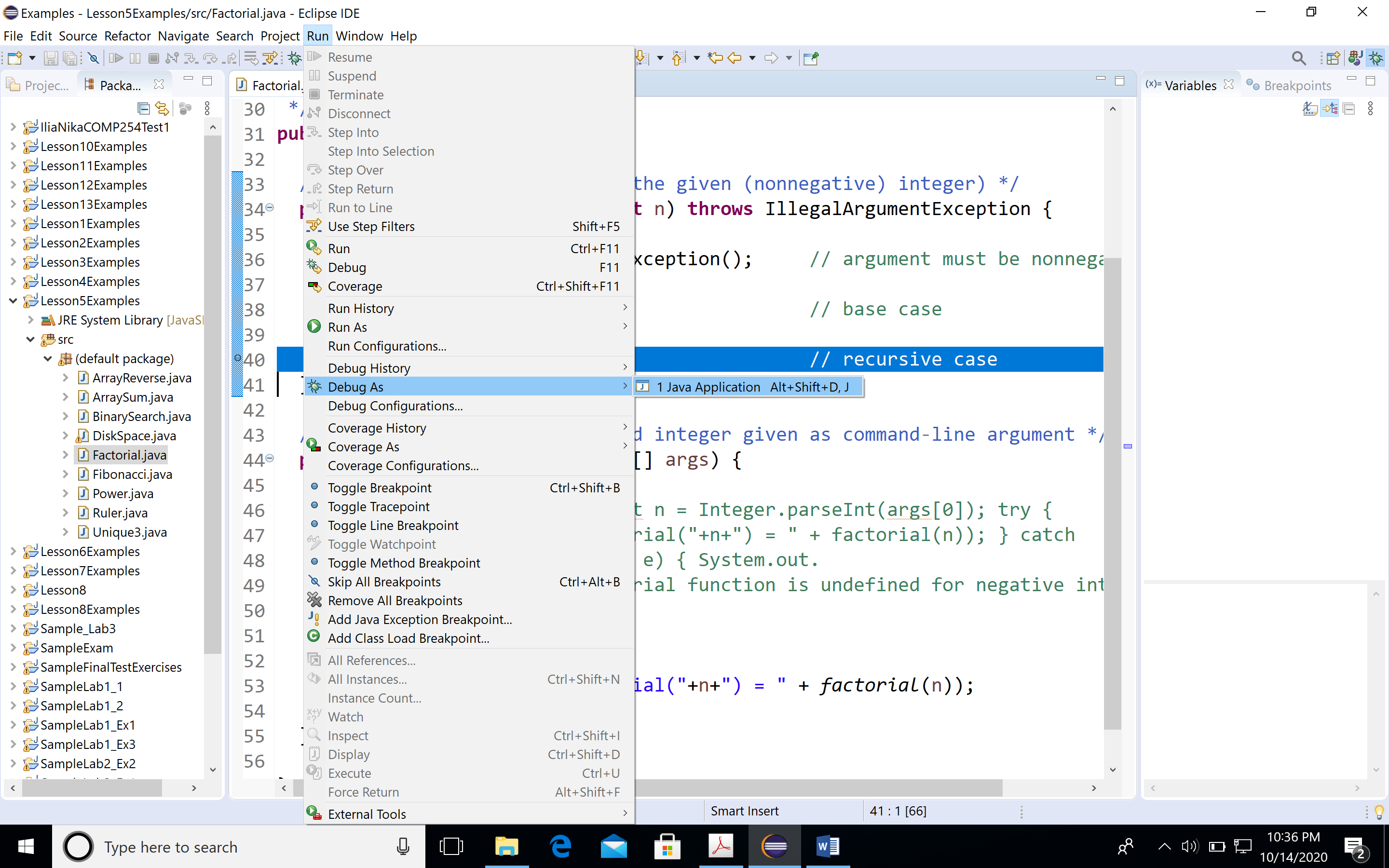
**Exercise 1: Recursive call stack**

In this exercise you will use the Debug tool in Eclipse IDE to view the call stack of a recursive method.

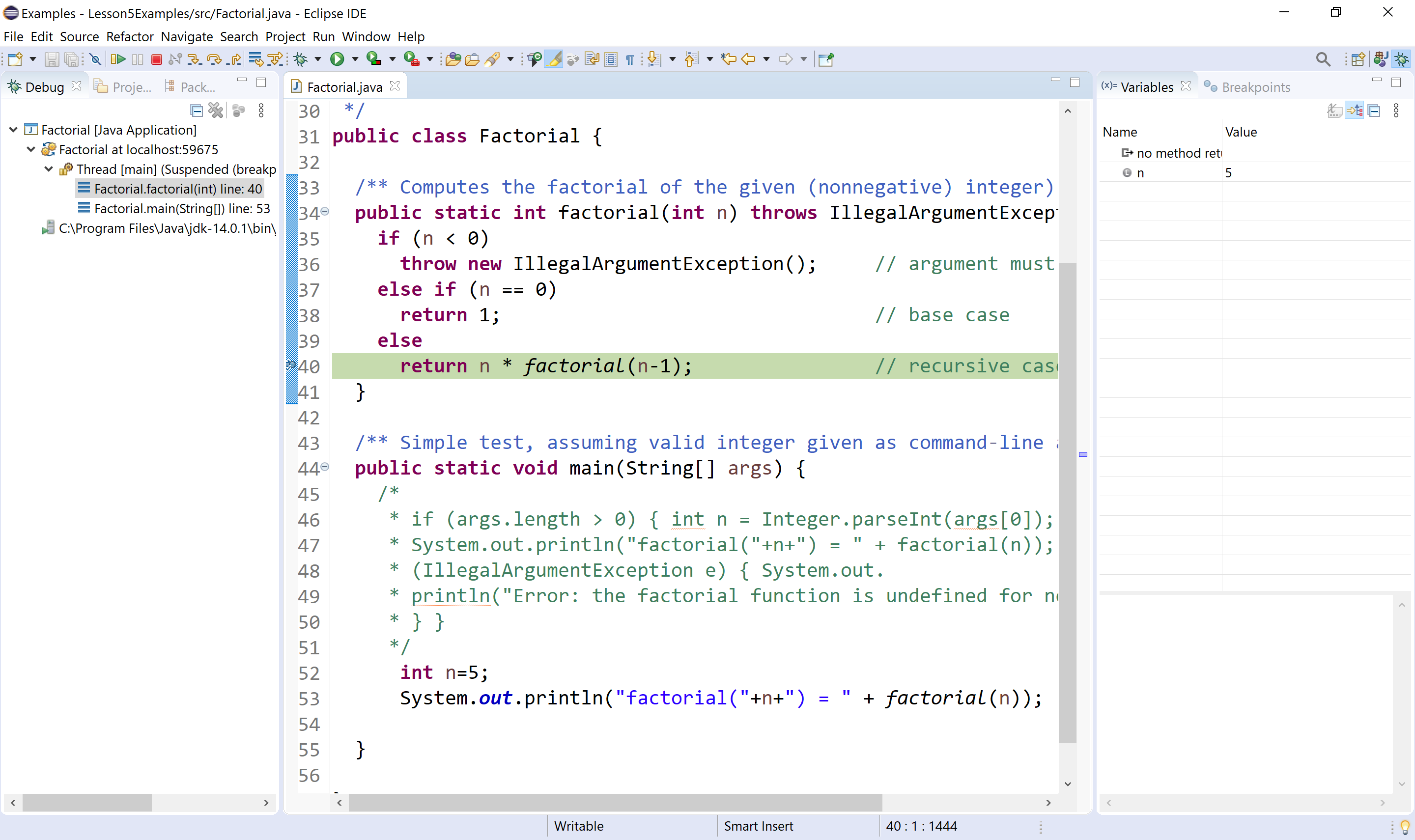
Open the Lesson5Examples project in Eclipse. Open the Factorial.java file. Set a breakpoint to line 40 (**return** n \* *factorial*(n-1);)



Run the debug tool as shown below:



Select Run menu, then Debug As Java Application.



Note the call stack on the left pane.

The call stack contains the following information:

* Arguments passed to *factorial* method
* Local variables of the method ( local variable is **n**)
* The line number where the method returns

Note the value of variable **n** in Variables window. The value of **n** is 5.

Press F6 to step over and check again the value of **n**. The value of **n** is 4.

Keep pressing F6 until the method returns and capture the values of **n** and the return values.

Explain the steps.

**Exercise 2: Maximum Element in an Array**

Let ***a*** be an array of integers. The length of array is ***n***. Using the same technique as the one used in *linearSum* algorithm from Lesson5Examples, write a Java method *maxElement*, that uses a recursive algorithm to find the maximum element in ***a***. Feel free to use **Math.max** function from Math class.

Visualize the call stack of the *maxElement* method as described in exercise 1.

**Exercise 3: Recursion Trace**

1. Draw the recursion trace for the execution of *maxElement* from exercise 2 on array *a* = 10, -2, 0, 7, 11, 8, 5, 6. -5, 13.
2. Draw the recursion trace for the execution of *binarySearch* method from Lesson5Examples on array *a* = 2, 7, 8, 12, 14, 17, 21, 23, 29, 35, 37, 38, 39, 41, 45. Use target = 37.

Use the same format as in Lesson 5 slides.

**Exercise 4: Performance of Recursive Algorithms**

Determine the performance of *maxElement* algorithm in terms of **O( )** notation.