**Debugging a Program in NetBeans**

**1. Write a Java Program**

The following program computes and prints the factorial of n (=1\*2\*3\*...\*n). The program, however, has a logical error and will produce a wrong answer for n=20 ("The Factorial of 20 is -2102132736" - a negative number?!).

|  |
| --- |
| /\*\* Compute the factorial of n \*/public class Factorial { // Print factorial of n public static void main(String[] args) { int n = 20; int factorial = 1;  // n! = 1\*2\*3...\*n for (int i = 1; i <= n; i++) { factorial \*= i; } System.out.println("The Factorial of " + n + " is " + factorial); }} |

Let us use the graphic debugger to debug the program.

**Not sure what a Factorial is? See below box:**

|  |
| --- |
| ***Factorials! Help****Example:****4!****is shorthand for****4 x 3 x 2 x 1***The **factorial function** (symbol: **!**) says to **multiply all whole numbers** from our chosen number down to 1.**Examples:****4!** = 4 × 3 × 2 × 1 = 24**7!** = 7 × 6 × 5 × 4 × 3 × 2 × 1 = 5040**1!** = 1 |

**2. Set an initial Breakpoint**

A breakpoint suspends program execution for you to examine the internal states of the program. Before starting the debugger, you need to set at least one breakpoint to suspend the execution inside the program. Set a breakpoint at main() method by clicking on the left-margin of the line containing main(). A red circle or an inverted Triangle appears in the left-margin indicating a breakpoint is set at that line.



**2. Start Debugging**

Right click anywhere on the source code ⇒ "Debug File". The program begins execution but suspends its operation at the breakpoint, i.e., the main() method.

As illustrated in the following diagram, the highlighted line (also pointed to by a green arrow) indicates the statement to be executed in the next step.



**3. Step-Over and Watch the Variables and Outputs**

Click the "Step Over" button (or select "Step Over" in "Debug" menu) to single-step thru your program. At each of the step, examine the value of the variables (in the "Variable" panel) and the outputs produced by your program (in the "Output" Panel), if any.

You can also place your cursor at any variable to inspect the content of the variable.



Single-stepping thru the program and watching the values of internal variables and the outputs produced is the ultimate mean in debugging programs - because it is exactly how the computer runs your program!

**Step 4: Breakpoint, Run-To-Cursor, Continue and Finish**

As mentioned, a breakpoint suspends program execution and let you examine the internal states of the program. To set a breakpoint on a particular statement, click on the left-margin of that line (or select "Toggle Breakpoint" from "Run" menu).

"Continue" resumes the program execution, up to the next breakpoint, or till the end of the program.

"Single-step" thru a loop with a large count is time-consuming. You could set a breakpoint at the statement immediately outside the loop (e.g., Line 11 of the above program), and issue "Continue" to complete the loop.

Alternatively, you can place the cursor on a particular statement, and issue "Run-To-Cursor" to resume execution up to the line.

"Finish" ends the debugging session. Always terminate your current debugging session using "Finish" or "Continue" till the end of the program.

**Other Debugger's Features:**

**Modify the Value of a Variable**

You can modify the value of a variable by entering a new value in the "Variable" panel. This is handy for temporarily modifying the behaviour of a program, without changing the source code.

**Step-Into and Step-Out**

To debug a method, you need to use "Step-Into" to step into the first statement of the method. You could use "Step-Out" to return back to the caller, anywhere within the method. Alternatively, you could set a breakpoint inside a method.