NetBeans For Java
How To Install and Get Started with Java Programming

1. How to Install NetBeans 8.2

Step 0: Install JDK

To use NetBeans for Java programming, you need to first install Java Development Kit (JDK). See "JDK - How to Install".

Step 1: Download

Download "NetBeans IDE" installer from http://netbeans.org/downloads/index.html. There are many "bundles" available. For beginners, choose the 1st entry "Java SE" (e.g., "`netbeans-8.2-javase-windows.exe`" 95MB).

Step 2: Run the Installer

Run the downloaded installer.

2. Writing a Hello-world Java Program in NetBeans

Step 0: Launch NetBeans

Launch NetBeans. If the "Start Page" appears, close it by clicking the "cross" button next to the "Start Page" title.

Step 1: Create a New Project

For each Java application, you need to create a "*project*" to keep all the source files, classes and relevant resources.

1. From "File" menu ⇒ Choose "New Project...".
2. The "Choose Project" dialog pops up ⇒ Under "Categories", choose "Java" ⇒ Under "Projects", choose "Java Application" ⇒ "Next".
3. The "Name and Location" dialog pops up ⇒ Under "Project Name", enter "`FirstProject`" ⇒ In "Project Location", select a suitable directory to save your works ⇒ Uncheck "Use Dedicated Folder for Storing Libraries" ⇒ **Uncheck "Create Main class"** ⇒ Finish.

Step 2: Write a Hello-world Java Program

1. Right-click on "`FirstProject`" ⇒ New ⇒ Java Class (OR choose the "File" menu ⇒ "New File..." ⇒ Categories: "Java", File Types: "Java Class" ⇒ "Next").
2. The "Name and Location" dialog pops up ⇒ In "Class Name", enter "`Hello`" ⇒ Delete the content in "Package" if it is not empty ⇒ "Finish".
3. The source file "`Hello.java`" appears in the editor panel. Enter the following codes:

```
public class Hello {
    public static void main(String[] args) {
        System.out.println("Hello, world");
    }
}
```

Step 3: Compile & Execute

There is no need to "compile" the source code in NetBeans explicitly, as NetBeans performs the so-called *incremental compilation* (i.e., the source statement is compiled as and when it is entered).

To run the program, right-click anywhere in the source (or from the "Run" menu) ⇒ Run File. Observe the output on the output console.

3. Debugging Program in NetBeans

Step 0: Write a Java Program

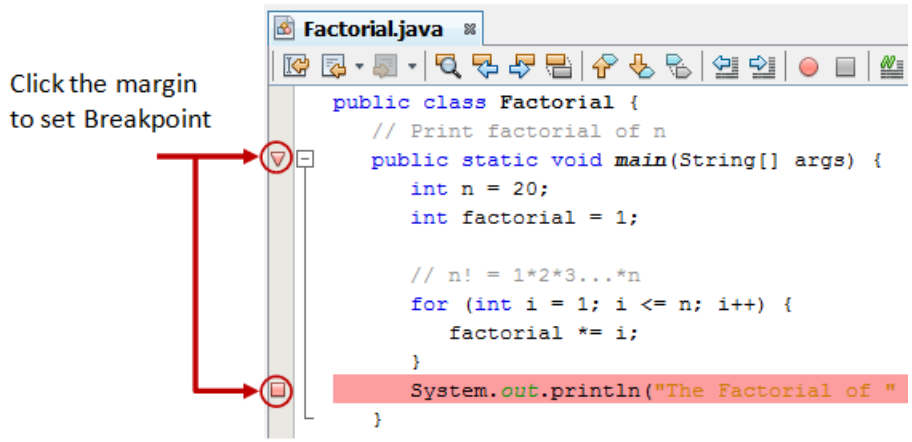
The following program computes and prints the factorial of *n* (=`1*2*3*...*`*n*). The program, however, has a logical error and produce a wrong answer for *n*=`20` ("`The Factorial of 20 is -2102132736`" - a negative number?!).

```
1/** Compute the factorial of n */
2public class Factorial {
3   // Print factorial of n
4   public static void main(String[] args) {
5      int n = 20;
6      int factorial = 1;
7
8      // n! = 1*2*3...*n
9      for (int i = 1; i <= n; i++) {
10        factorial *= i;
11     }
12     System.out.println("The Factorial of " + n + " is " + factorial);
13  }
14}
```

Let us use the graphic debugger to debug the program.

Step 1: Set an initial Breakpoint

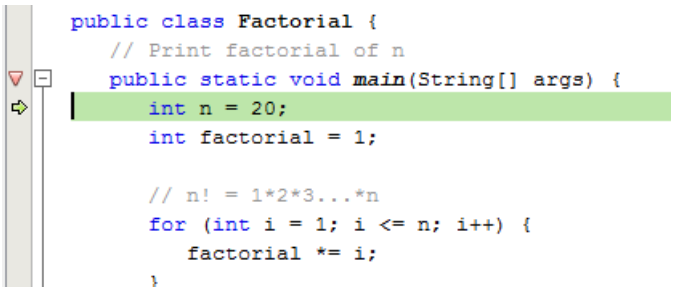
A *breakpoint* suspends program execution for you to examine the internal states of the program. Before starting the debugger, you need to set at least one breakpoint to suspend the execution inside the program. Set a breakpoint at `main()` method by clicking on the *left-margin* of the line containing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



Step 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.

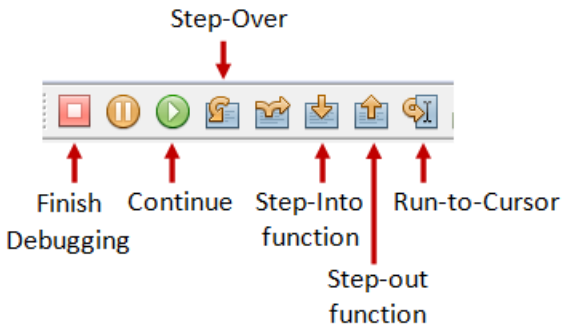
As illustrated in the following diagram, the highlighted line (also pointed to by a green arrow) indicates the statement to be executed in the *next* step.

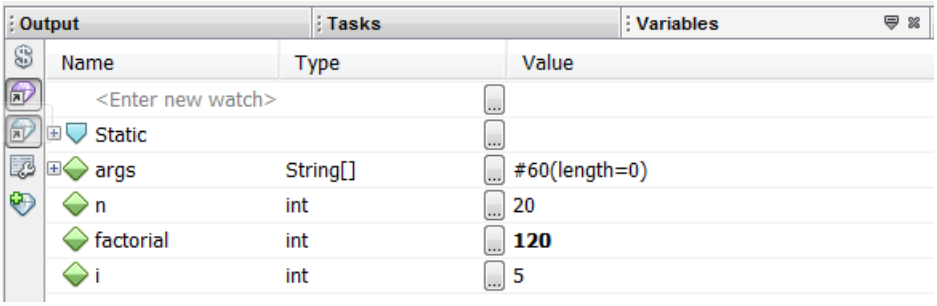


Step 3: Step-Over and Watch the Variables and Outputs

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.

3.1 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.