if(a<b){ //line1

System.out.println("frefe"); //line2

} //line3

System.out.println("sefwefrefe"); //line4

* Normally we can define a formula like N(S+C) to calculate a complexity of a simple code.
* There N is nesting level, S is size which calculates the count line wise basically program statement. We can ignore operators like (), {}, ; and etc as they are depending upon the programming language even. And in the above formula “C” is the control structures like for, if and etc where weights can be given 3,2 and etc respectively because in “**If”** there is only to check one or more conditions but in a loop like “**for”** there may have condition plus the initializations and increments. And also we should multiply the complexity inside the loop by number of iterations.
* I considered the count of “System.out.println” as one because it’s like one program keyword to print the output.

When the condition gets **true**,

|  |  |  |  |
| --- | --- | --- | --- |
| line | Size(S) | Control Structures(C) | Complexity N(S+C) |
| 1 | 3 | 2 | 1(3+2) = 5 |
| 2 | 2 | 0 | 2(2+0) = 4 |
| 3 | 0 | 0 | 0 |
| 4 | 2 | 0 | 1(2+0) = 2 |

Total complexity when it’s true = 5+4+0+2 = 11

When the condition gets **false**,

|  |  |  |  |
| --- | --- | --- | --- |
| line | Size(S) | Control Structures(C) | Complexity N(S+C) |
| 1 | 3 | 2 | 1(3+2) = 5 |
| 2 | 0 | 0 | 2(0+0) = 0 |
| 3 | 0 | 0 | 0 |
| 4 | 2 | 0 | 1(2+0) = 2 |

Total complexity when it’s false = 5+0+0+2 = 7

By considering both true and false total complexity can be derived using the **Median** of both true and false

**Total complexity**= (11+7)/2 = 9