**Recursion**

A method is a recursive method when it calls itself.

**Factorial of a number**

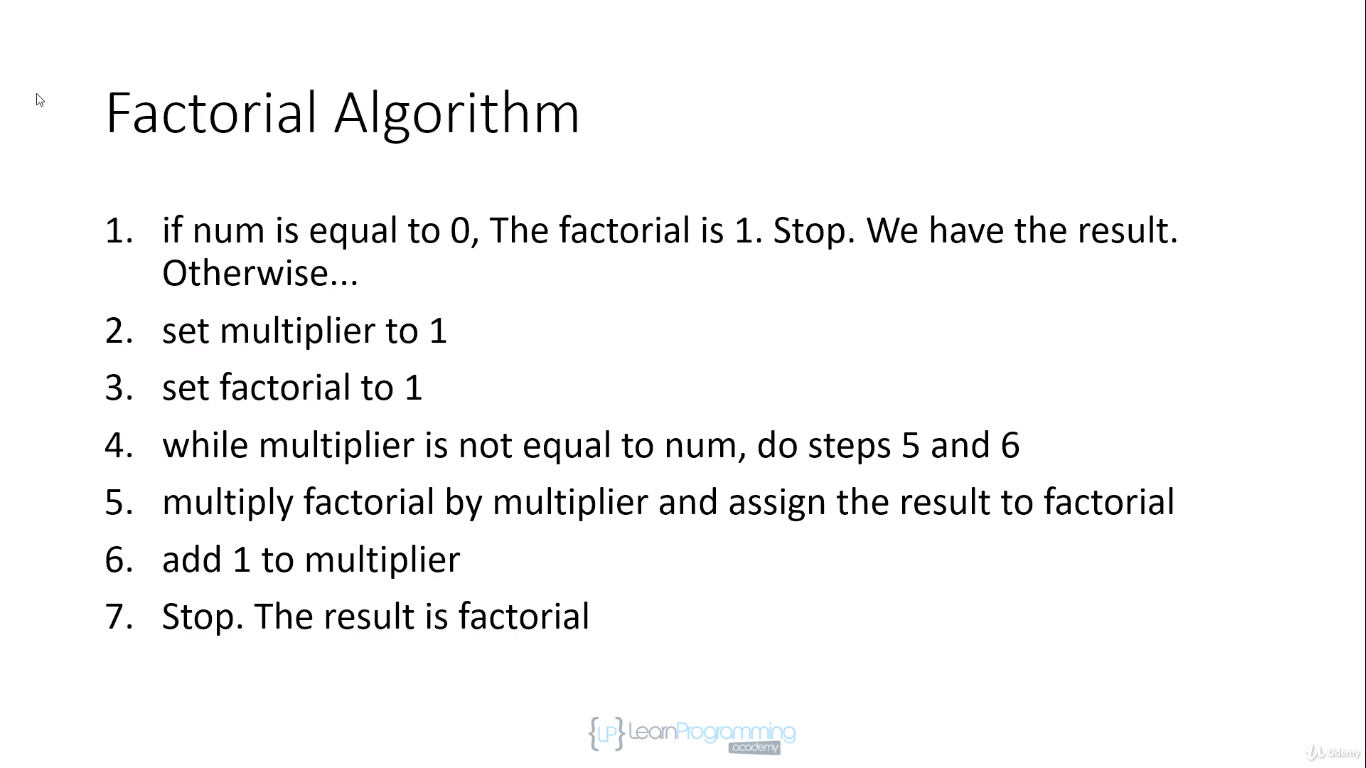
3! = 3\*2\*1 OR 3 \*2!

2! = 2\*1 OR 2\*1!

1! =1 OR 1\*0!

0! =1

**Iterative Solution for factorial**



Factorial for -ve number is undefined.

**Factorial Solution**

1! = 1 = 1 \* 0!

2! = 2\*1 = 2 \* 1!

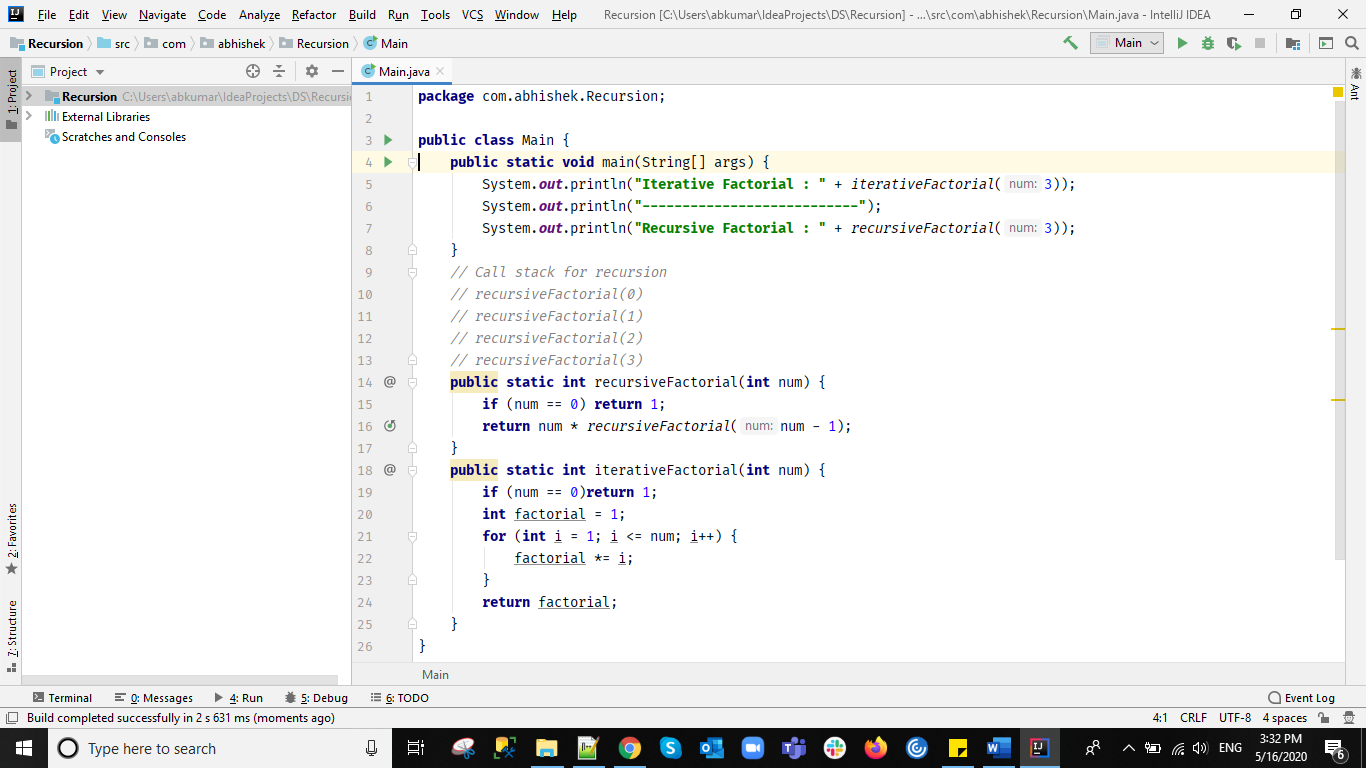
3! = 3\*2\*1 = 3 \* 2!

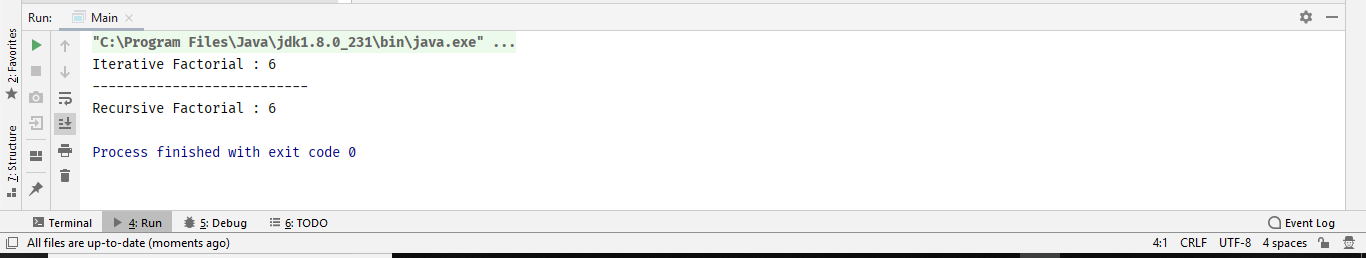
4! = 4\*3\*2\*1 = 4 \* 3!

5! = 5\*4\*3\*2\*1 = 5 \* 4!

N! = N \* (N-1)!

*// Call stack for recursion  
// recursiveFactorial(0)  
// recursiveFactorial(1)  
// recursiveFactorial(2)  
// recursiveFactorial(3)*





*Recursive call ends when there is end/Break/base condition, because recursiveFactorial(3) will only return when recursiveFactorial(2) will return result*

*Recursive call may through stackoverflowerror when stack fulls calling recursivemethod again and again.*

*If we don’t have breakcondition it will throw stackoverflowerror as it blow callstack.*

*Iterativeway takes less memory corresponding to recursion.*

*Call Stack/Recursion Stack can get stackoverflowerror even if you have breaking condition as it may call more than million times and not reaching end condition.*

*Solution is tail recursion. Java doesn’t use tail condition.*