**Java Recursion**

In Java, a method that calls itself is known as a recursive method. And, this process is known as recursion.

A physical world example would be to place two parallel mirrors facing each other. Any object in between them would be reflected recursively.

### How Recursion works?

|  |
| --- |
| A function is calling itself |
| Working of Java Recursion |

In the above example, we have called the recurse() method from inside the main method. (normal method call). And, inside the recurse() method, we are again calling the same recurse method. This is a recursive call.

In order to stop the recursive call, we need to provide some conditions inside the method. Otherwise, the method will be called infinitely.

Hence, we use the [if...else statement](https://www.programiz.com/java-programming/if-else-statement) (or similar approach) to terminate the recursive call inside the method.

## Example: Factorial of a Number Using Recursion

class Factorial {

static int factorial( int n ) {

if (n != 0) // termination condition

return n \* factorial(n-1); // recursive call

else

return 1;

}

public static void main(String[] args) {

int number = 4, result;

result = factorial(number);

System.out.println(number + " factorial = " + result);

}

}

**Output**:

4 factorial = 24

In the above example, we have a method named factorial(). The factorial() is called from the main() method. with the number variable passed as an argument.

Here, notice the statement,

return n \* factorial(n-1);

The factorial() method is calling itself. Initially, the value of n is 4 inside factorial(). During the next recursive call, 3 is passed to the factorial() method. This process continues until n is equal to 0.

When n is equal to 0, the if statement returns false hence 1 is returned. Finally, the accumulated result is passed to the main() method.

## Working of Factorial Program

The image below will give you a better idea of how the factorial program is executed using recursion.

|  |
| --- |
| Finding the factorial of a number using recursion |
| Factorial Program using Recursion |

## Advantages and Disadvantages of Recursion

When a recursive call is made, new storage locations for variables are allocated on the stack. As, each recursive call returns, the old variables and parameters are removed from the stack. Hence, recursion generally uses more memory and is generally slow.

On the other hand, a recursive solution is much simpler and takes less time to write, debug and maintain.