



**public** **class** SumOfNumbers {

**static** **int** *result* = 0;

**public** **static** **int** sumUsingIterative(**int** n) {

**for** (**int** i = 1; i <= n; i++) {

*result* += i;

}

**return** *result*;

}

**public** **static** **int** sumUsingRecursion(**int** n) {

**if** (n == 1) {

**return** 1;

}

**return** n + *sumUsingRecursion*(n - 1);

}

}

**public** **class** AppTest {

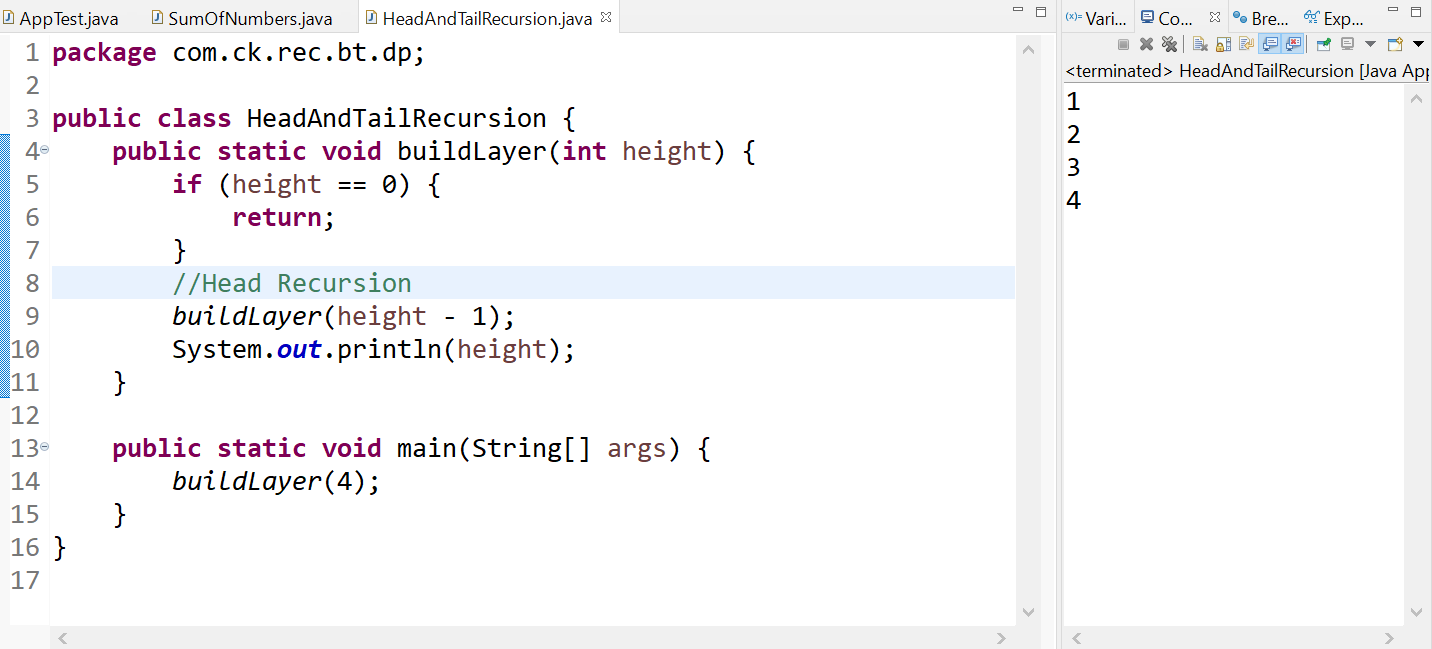
**public** **static** **void** main(String[] args) {

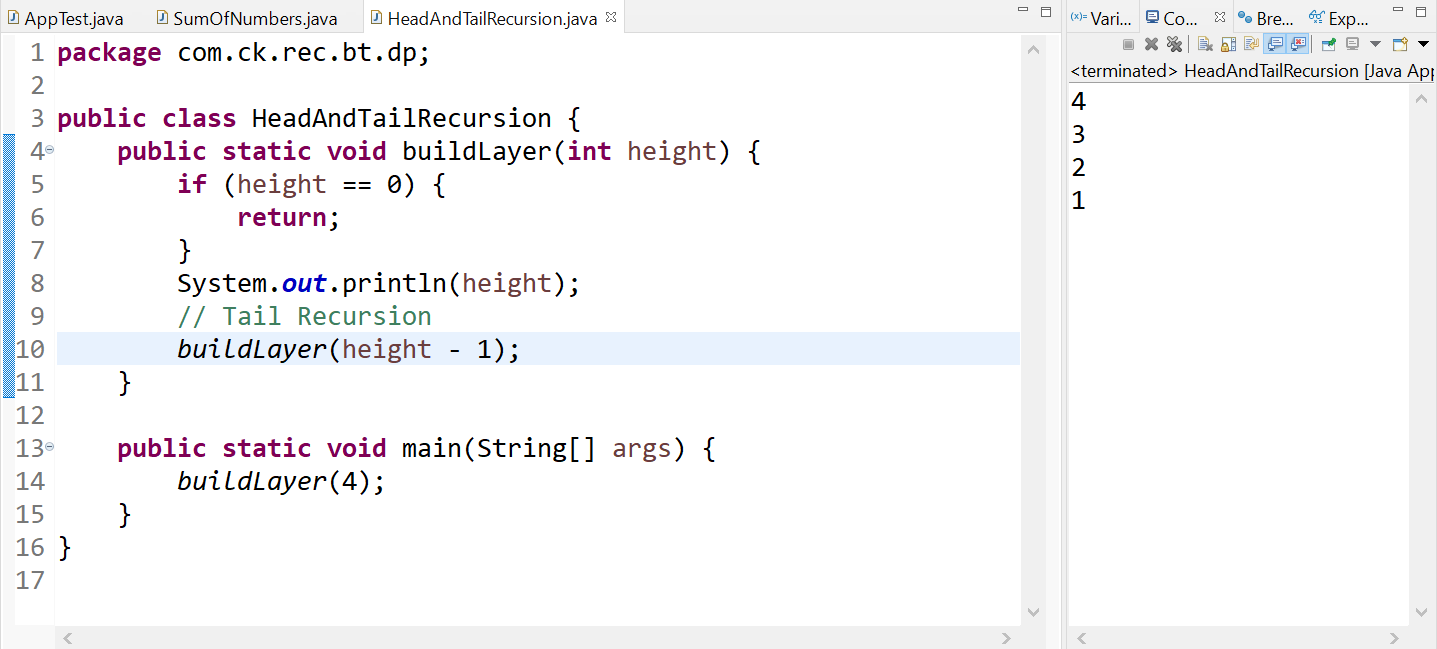
System.***out***.println(SumOfNumbers.*sumUsingIterative*(5));

System.***out***.println(SumOfNumbers.*sumUsingRecursion*(5));

}

}





* Tail Recursion is very similar to Iteration (similar to **while** or **for** loop)
* If Head recursion, we need to use OS stack to pile up the method calls and arguments.

**public** **class** Factorial {

**public** **static** **int** factorial(**int** n) {

**if** (n == 1) {

**return** 1;

}

**return** n \* *factorial*(n - 1);

}

**public** **static** **void** main(String[] args) {

System.***out***.println(*factorial*(5));

}

}

For the above problem, as discussed previously, there are two steps

1. Pushing into stack
2. Popping the result and performing multiplication.

If we see, step two is just returning the result. This can be optimized by using **accumulator**.

**public** **class** FactorialOptimized {

**public** **static** **int** factorial(**int** accumulator, **int** n) {

**if** (n == 1) {

**return** accumulator;

}

**return** *factorial*(accumulator \* n, n - 1);

}

**public** **static** **int** calculateFactorial(**int** n) {

**return** *factorial*(1, n);

}

**public** **static** **void** main(String[] args) {

System.***out***.println(*calculateFactorial*(5));

}

}