**Using a Mathematical Formula (Efficient Approach)**

The sum of the squares of the first n natural numbers is given by:

S=n(n+1)(2n+1)6S = \frac{n(n + 1)(2n + 1)}{6}S=6n(n+1)(2n+1)​

This formula allows us to compute the result in constant time **O(1)**.

java

نسختحرير

public class SquareSumCalculator {

public static int sumOfSquares(int n) {

if (n < 1) {

throw new IllegalArgumentException("n must be a positive integer.");

}

return (n \* (n + 1) \* (2 \* n + 1)) / 6;

}

public static void main(String[] args) {

System.out.println(sumOfSquares(3)); // Output: 14 (1² + 2² + 3² = 1 + 4 + 9)

System.out.println(sumOfSquares(5)); // Output: 55 (1² + 2² + 3² + 4² + 5²)

System.out.println(sumOfSquares(1)); // Output: 1 (1²)

}

}

**Using a Loop (Iterative Approach)**

If you prefer a more straightforward approach using iteration:

java

نسختحرير

public class SquareSumCalculator {

public static int sumOfSquares(int n) {

if (n < 1) {

throw new IllegalArgumentException("n must be a positive integer.");

}

int sum = 0;

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

sum += i \* i;

}

return sum;

}

public static void main(String[] args) {

System.out.println(sumOfSquares(3)); // Output: 14

System.out.println(sumOfSquares(5)); // Output: 55

System.out.println(sumOfSquares(1)); // Output: 1

}

}

**Comparison:**

| **Method** | **Time Complexity** | **Space Complexity** | **Notes** |
| --- | --- | --- | --- |
| **Formula** | O(1) | O(1) | Most efficient |
| **Loop** | O(n) | O(1) | Easier to understand |