**Using a Loop (Iterative Approach)**

public class OddSumCalculator {

public static int sumOddUpTo(int n) {

if (n < 1) {

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

}

int sum = 0;

for (int i = 1; i <= n; i += 2) { // Increment by 2 to get only odd numbers

sum += i;

}

return sum;

}

public static void main(String[] args) {

System.out.println(sumOddUpTo(5)); // Output: 9 (1 + 3 + 5)

System.out.println(sumOddUpTo(10)); // Output: 25 (1 + 3 + 5 + 7 + 9)

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

}

}

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

The sum of the first k odd numbers is given by:

S=k2S = k^2S=k2

where k is the count of odd numbers up to n (i.e., k = (n + 1) / 2).

public class OddSumCalculator {

public static int sumOddUpTo(int n) {

if (n < 1) {

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

}

int k = (n + 1) / 2; // Number of odd numbers up to n

return k \* k; // Sum of first k odd numbers

}

public static void main(String[] args) {

System.out.println(sumOddUpTo(5)); // Output: 9 (1 + 3 + 5)

System.out.println(sumOddUpTo(10)); // Output: 25 (1 + 3 + 5 + 7 + 9)

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

}

}

**Comparison:**

| **Method** | **Time Complexity** | **Space Complexity** | **Notes** |
| --- | --- | --- | --- |
| **Loop** | O(n) | O(1) | Simple but slower for large n |
| **Formula** | O(1) | O(1) | More efficient |