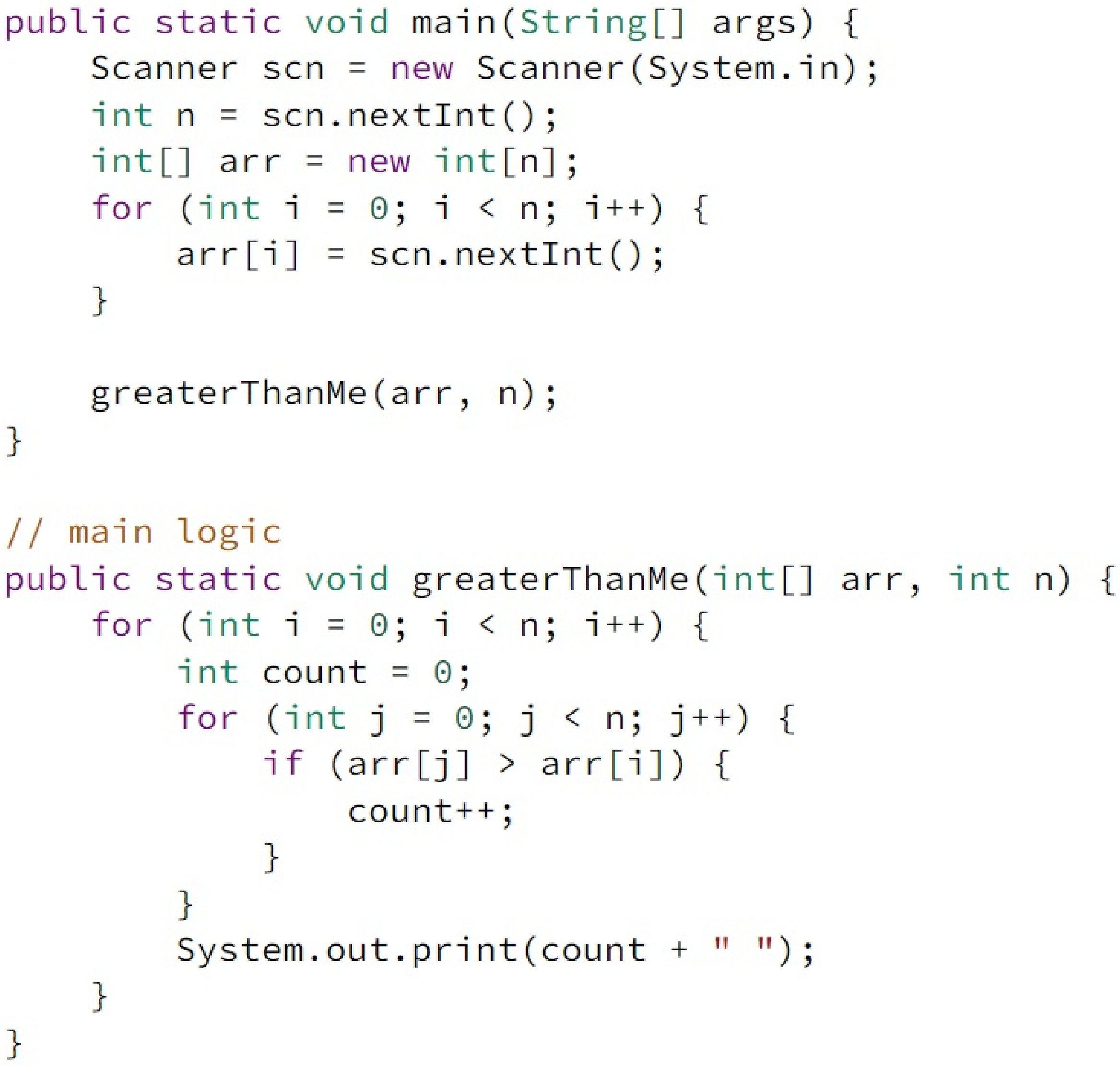
Code



```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    greaterThanMe(arr, n);
}

// main logic
public static void greaterThanMe(int[] arr, int n) {
    for (int i = 0; i < n; i++) {
        int count = 0;
        for (int j = 0; j < n; j++) {
            if (arr[j] > arr[i]) {
                count++;
            }
        }
        System.out.print(count + " ");
    }
}
```

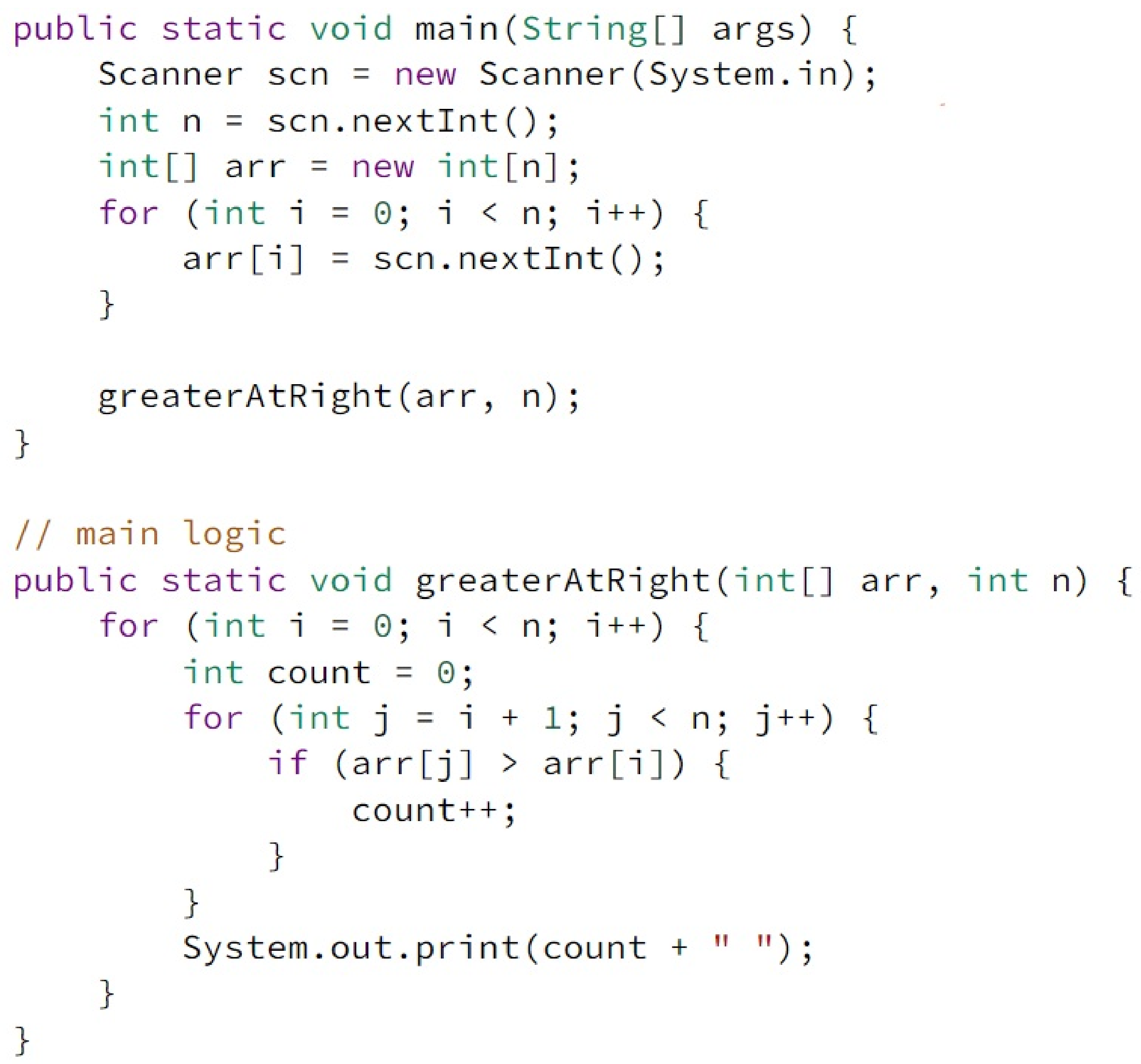
n = 4

5	0	3	2
0	1	2	3

i=0, j=0	5>5	false	C=0
j=1	0>5	false	
j=2	3>5	false	
j=3	2>5	false	
i=1, j=0	5>0	true	C=3
j=1	0>0	false	
j=2	3>0	true	
j=3	2>0	true	
i=2, j=0	5>3	true	C=1
j=1	0>3	false	
j=2	3>3	false	
j=3	2>3	false	
i=3, j=0	5>2	true	C=2
j=1	0>2	false	
j=2	3>2	true	
j=3	2>2	false	

Greater At Right

Code



```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    greaterAtRight(arr, n);
}

// main logic
public static void greaterAtRight(int[] arr, int n) {
    for (int i = 0; i < n; i++) {
        int count = 0;
        for (int j = i + 1; j < n; j++) {
            if (arr[j] > arr[i]) {
                count++;
            }
        }
        System.out.print(count + " ");
    }
}
```

combinat^n
without
repetation

maximum difference between the two elements

faith :- j^{th} element will be greater and i^{th} element will be smaller

condition :-

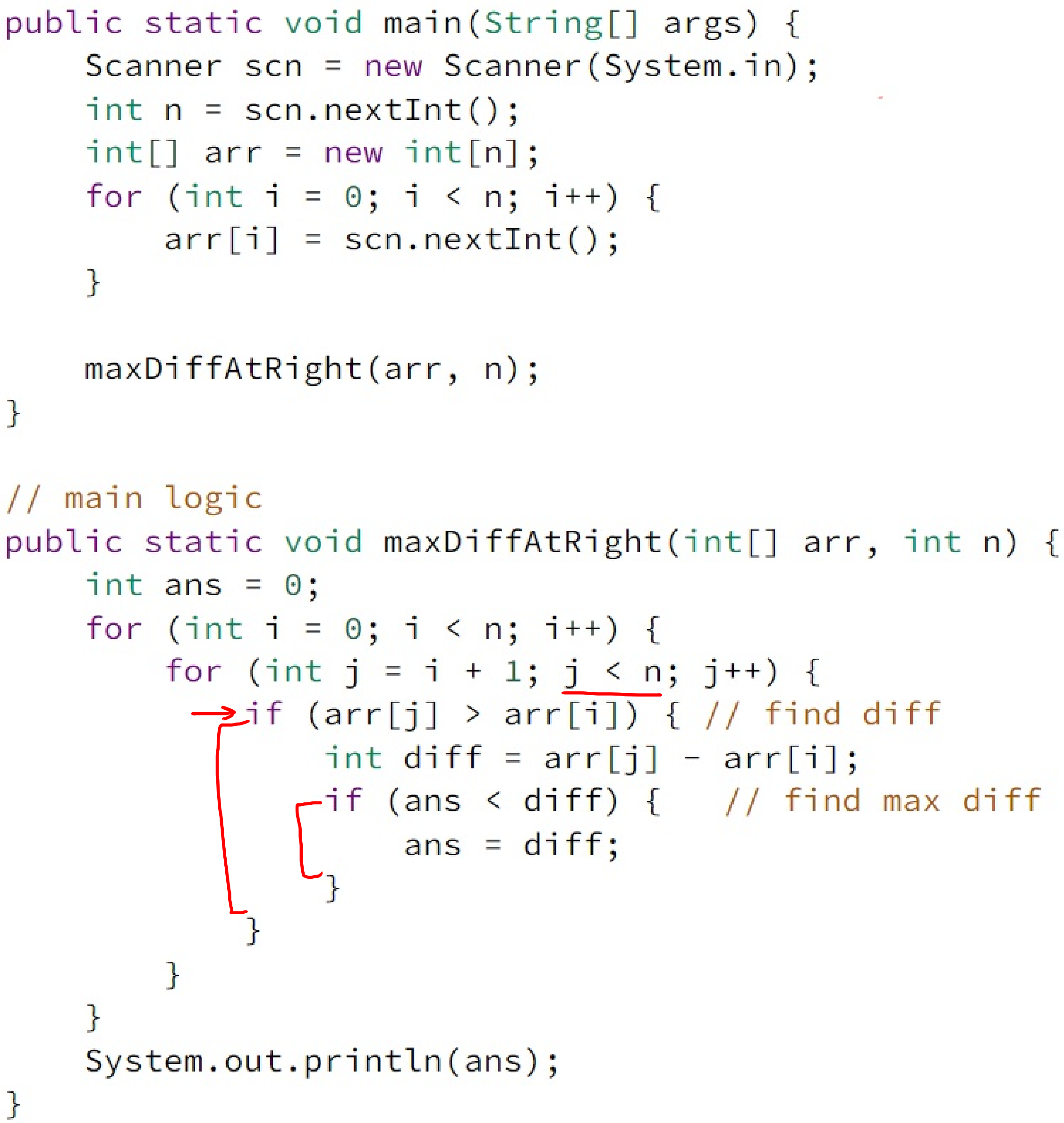
larger element should appear after the smaller one only

n = 7

2	3	10	6	4	8	1
0	1	2	3	4	5	6

diff = 0

i = 0, j = 1	3 > 2	diff = 1
j = 2	10 > 2	diff = 8
j = 3	6 > 2	diff = 8
j = 4	4 > 2	diff = 8
j = 5	8 > 2	diff = 8
j = 6	1 > 2	diff = 8



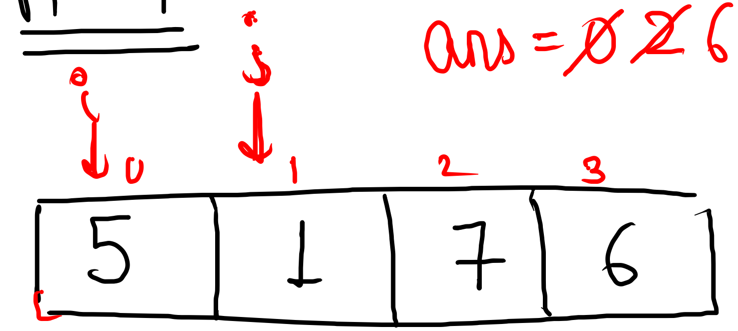
```java
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    int[] arr = new int[n];
    for (int i = 0; i < n; i++) {
        arr[i] = scn.nextInt();
    }

    maxDiffAtRight(arr, n);
}

// main logic
public static void maxDiffAtRight(int[] arr, int n) {
    int ans = 0;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (arr[j] > arr[i]) { // find diff
                int diff = arr[j] - arr[i];
                if (ans < diff) {   // find max diff
                    ans = diff;
                }
            }
        }
    }
    System.out.println(ans);
}
```

n = 4

ans = ~~0~~ ~~2~~ 6



i= 0, j=1   1>5

j=2   7>5   diff = 2

j=3   6>5   diff = 1

i=1, j=2   7>1   diff = 6

j=3   6>1   diff = 5

i=2 j=3   6>7

i=3, j=4   4<4 →