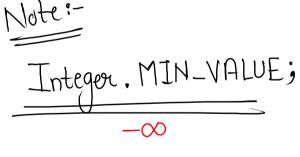
Note:
if a function is of return type, then it must return something all the time

Maximum of Array

$$N = 6$$
 $OUT = \begin{bmatrix} -2 & 3 & 1 & 4 & 2 & 1 \\ 6 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}$



- 1) create a variable max = $-\infty$
- 2) traverse in avoidy from start to end 2.1) check if coverent value > max
 - 2.1.1) max = courent value
- 3) return mar

```
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();
   int ans = maxOfArray(arr);
    System.out.println(ans);
public static int maxOfArray(int[] arr) {
→int max = Integer.MIN VALUE;
\rightarrowint n = arr.length;
  for (int i = 0; i < n; i++) {</pre>
       rif ( arr[i] > max ) {
            max = arr[i];
    return max;
```

```
public static int maxOfArray(int[] arr) {
   int max = Integer.MIN_VALUE;
   int n = arr.length;
   for (int i = 0; i < n; i++) {
      max = Math.max( max, arr[i] );
   }
   return max;
}</pre>
```

$$n=6$$
 $avot = -2 \quad 3 \quad 1 \quad 4 \quad 2 \quad 1$
 $max = -60 \quad -2 \quad 3 \quad 4$
 $n = 6$
 $i=0, (-2 > -\infty)$ true

 $i=1, (3 > -2)$ true

 $i=2, (1 > 3)$ false

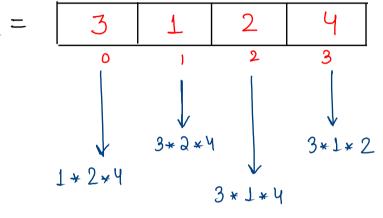
 $i=3, (4 > 3)$ true

 $i=4, (2 > 4)$ false

 $i=5, (1 > 4)$ false

 $i=6$

Hefore 4



$$Qn2 = 8$$
 24 12 6

(=2)

i=3,

- ma

$$0001 = 3 0 2$$
 $0001 = 3 \times 1 \times 2 \times 4$

$$i = 0, \quad ans = 1$$



an = prod

an [2]

$$\hat{l} = 1, \qquad \text{and} = \frac{prod}{prod}$$

$$curs = \frac{pro}{1}$$

approch 2) find prod, of all elements when indexes are not same

$$n=4$$

and $n=4$

and

indexes are not same

$$\frac{n=4}{2007} = 3 \quad 0 \quad 2 \quad 4$$

$$\frac{n^{2} + y}{0007} = \frac{3}{3} = \frac{3}{0} = \frac{3}{2} = \frac{3}{4}$$

$$= 0, \quad j = 0 \quad \text{to} \quad (n-1), \quad \text{if} \quad (i!=j) \quad \text{ans} = \text{ans} * \text{ans} [j]$$

$$= 1, \quad j = 0 \quad \text{to} \quad (n-1), \quad \text{if} \quad (i!=j) \quad \text{ans} = \text{ans} * \text{ans} [j]$$

$$= 2, \quad j = 0 \quad \text{to} \quad (n-1), \quad \text{if} \quad (i!=j) \quad \text{ans} = \text{ans} * \text{ans} [j]$$

$$i = 0, \quad j = 0 \text{ to } (n-1), \quad if (i!=j) \quad ans = ans * avr[j]$$

$$i = 1, \quad j = 0 \text{ to } (n-1), \quad if (i!=j) \quad ans = ans * avr[j]$$

$$i = 2, \quad j = 0 \text{ to } (n-1), \quad if (i!=j) \quad ans = ans * avr[j]$$

$$i = 3, \quad j = 0 \text{ to } (n-1), \quad if (i!=j) \quad ans = ans * avr[j]$$

$$i = 0$$
, $j = 0$,
 $j = 1$, and = and + 0
 $j = 2$, and = and + 0 * 2
 $j = 3$, and = and + 0 * 2 * 4
print and = 0

j = 2, an = an + 3 * 2

print ans = 24

j=3, an=an+3+2+4

on = 1

Our = 1

an = 1

$$j = 0$$
, $j = 0$,

 $j = 1$, an = an * 3

 $j = 1$, an = an * 3 * 0

 $j = 3$, an = an * 0 * 2 * 4

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3 * 0 * 9

 $j = 3$, an = an * 3

j = 1, an = an * 3 * 0

j=3,

point au = 0

j = 2, $an = an \times 3 \times 0 \times 2$

psudo

1) input array 2) traverse from start to end with i index 2.1) dedone au = 1 2.2) traverse from start to end with i index 2.2.1) check if (i != i) then aru = aru * avr[j] 2.3) print ans

```
-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();
       productExceptItself(arr, n);
 public static void productExceptItself(int[] arr, int n) {
      for (int i = 0; i < n; i++) {
   int ans = 1;
   for (int j = 0; j < n; j++) {
     if ( i != j ) { // except itself
        ans = ans * arr[j];
        }
}</pre>
            System.out.println(ans);
```