⇒ Comparator & Comparable (it will not effect T.C)

↳ used to modify the logic of inbuilt function

↳ Arrays.sort(arr)

Syntax :

code

```
public static void main(String[] args) {
    int[] arr = { 5, 6, 3, 1, -5, 8, 2 };
    int n = arr.length;
    Integer[] arr1 = new Integer[n];
    for (int i = 0; i < n; i++) {
        arr1[i] = arr[i];
    }
    // comparable & comparator
    Arrays.sort(arr1, new myComparator());

    for (int i = 0; i < n; i++) {
        System.out.print(arr1[i] + " ");
    }
}
```

declaration

implimentation

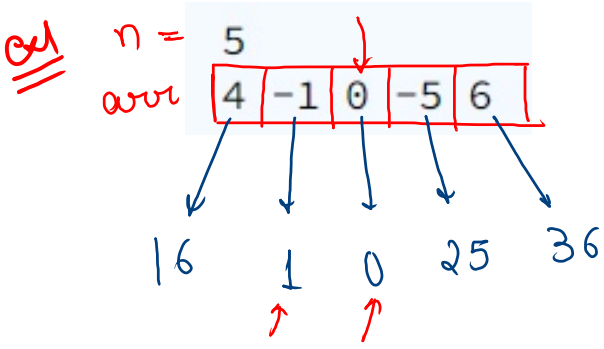
```
public static class myComparator implements Comparator<Integer> {
    @Override
    public int compare(Integer a, Integer b) {
        return a - b;  // increasing order
        // return b - a; // decreasing order
    }
}
```

Note:-

in compare f"

- if we return -ve (-1) value :- ascending
- if we return +ve (+1) value :- descending
- if we return "a-b" value :- ascending but acc. to values
- if we return "b-a" value :- desending but acc to values

Sort the array according to their Square of each element (ascending)



arr | 0 | -1 | 4 | -5 | 6 ✓

n = 7

ex2

	(49)	(9)	(1)	(16)	(4)	(64)	(81)
arr ⇒	-7	3	-1	4	2	8	-9

array after sorting

arr ⇒	-1	2	3	4	-7	8	-9

→ Comparable Comparator

⇒ Square of Sorting

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

    Arrays.sort(arr, new myComparator());

    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }
}

public static class myComparator implements Comparator<Integer> {
    @Override
    public int compare(Integer a, Integer b) {
        return a * a - b * b;
    }
}
```

O(N log N)

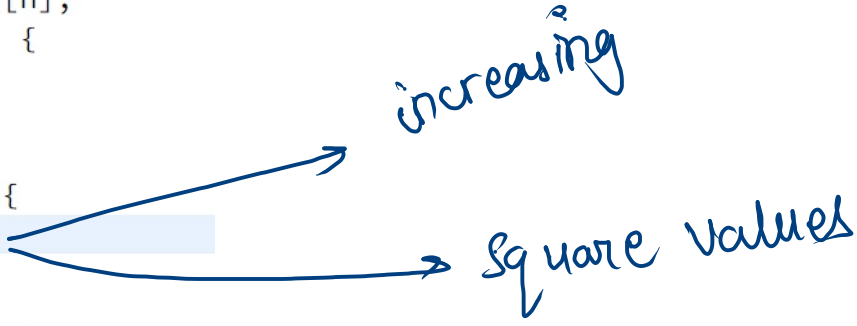
⇒ Lambda function // one line function

```
Arrays.sort(arr, (a,b)-> {
        return a-b;
});
```

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

    Arrays.sort(arr, (a, b) -> {
        return a * a - b * b;
    });

    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }
}
```



Ques Sort array in asc. order but acc. to cube

return a*a*a − b*b*b;

Sort Array By Parity

a = even
b = odd

n = 7

arr =

3	4	2	8	7	1	5

first all even values and then odd values

arr =

4	2	8	3	7	1	5

non-decreasing means increasing order

arr =

2	4	8	1	3	5	7

↳ logic a, b

8	3

↳ a = even, b = even → a-b

↳ a = even, b = odd → −1

↳ a = odd, b = odd → a-b

↳ a = odd, b = even → +1

Note :-

-ve → ascending (a → b)

+ve → descending (b → a)

Code

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

    // main logic
    Arrays.sort(arr, (a, b) -> {
        if ( a % 2 == 0 && b % 2 == 0 ) {
            return a - b;
        } else if ( a % 2 != 0 && b % 2 != 0 ) {
            return a - b;
        } else if ( a % 2 == 0 && b % 2 != 0 ) {
            return -1;
        } else {
            return 1;
        }
    });

    for (int i = 0; i < n; i++) {
        System.out.print(arr[i] + " ");
    }
}
```

1
2
3
4

Ques Sort the array acc.

↳ odd first, even last

↳ have odd in ↑ ing

↳ even ↓ ing

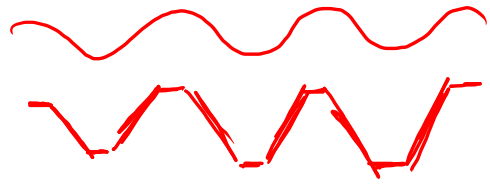
1 → b - a

2 → a - b

3 → #L → a - even, b - odd

4 → 4

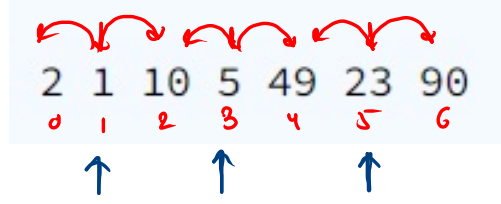
Sort an array in wave form 1



arr

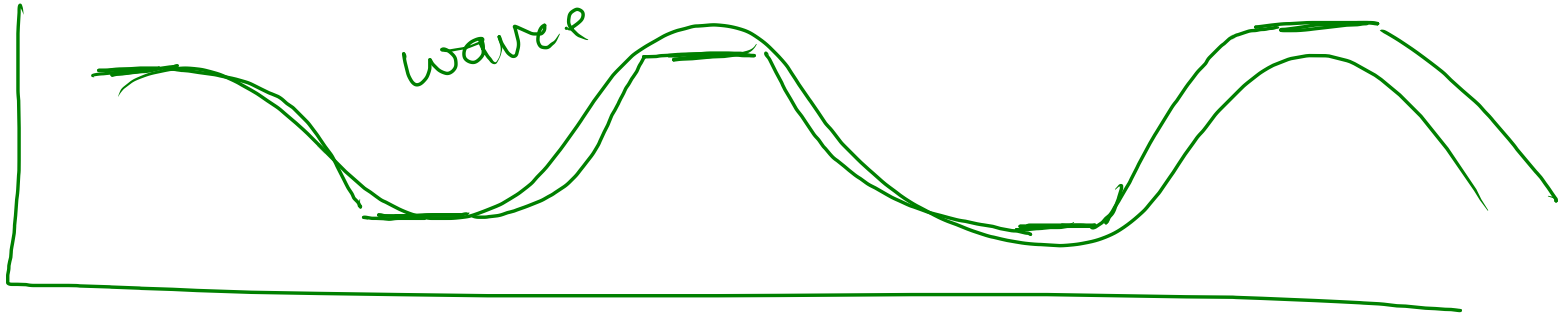
```
7
10 90 49 2 1 5 23
```

ans

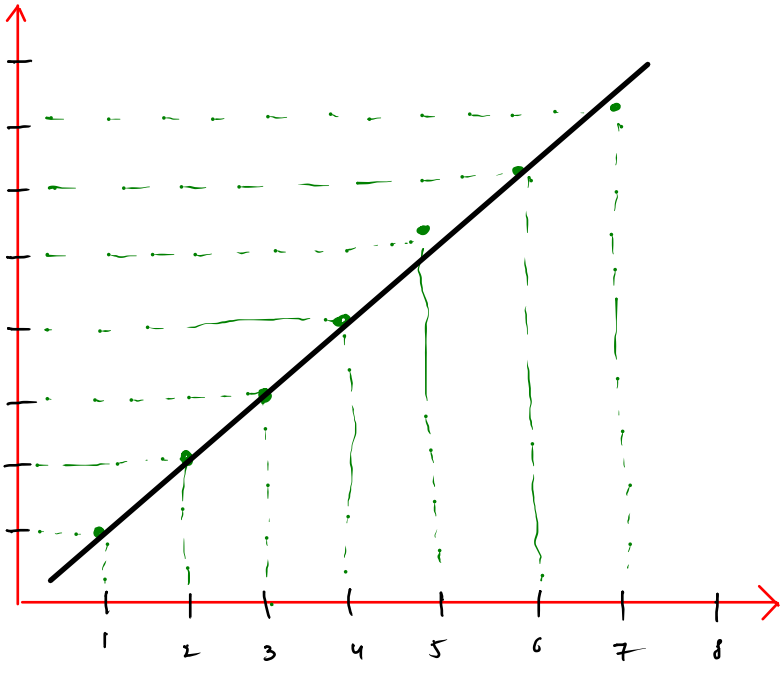


wave

```
arr[0] >= arr[1] <= arr[2] >= arr[3] <= arr[4] >= …..
```

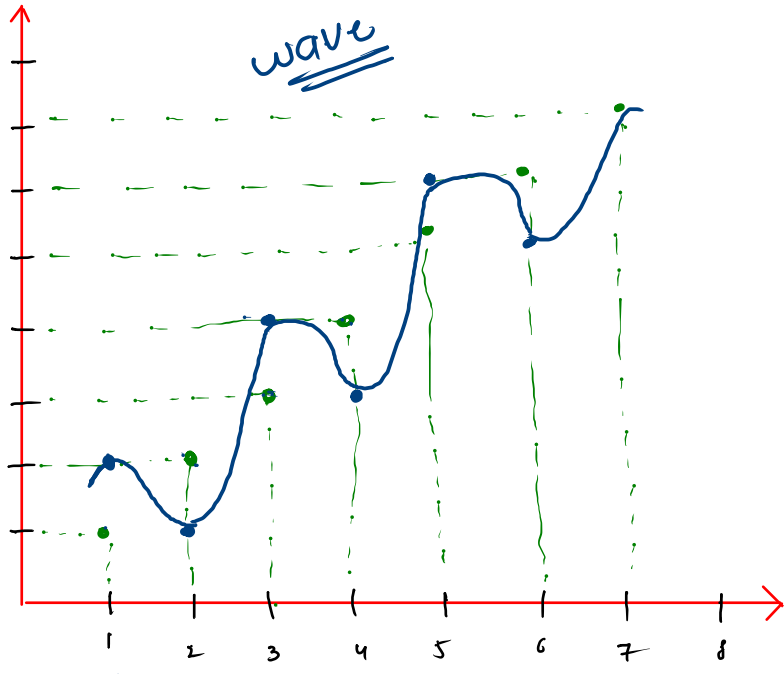


arr = [5, 1, 2, 7, 3, 6, 4]



1) Sort the array in ↑ing order

arr = [1, 2, 3, 4, 5, 6, 7]



2) Swap adj. values

arr = [2, 1, 4, 3, 6, 5, 7]

wave

code

```java
public static void waveForm(int[] arr) {
    // step1 : sort the array
    Arrays.sort(arr);

    // step2: swap adj. elements
    for (int i = 0; i < arr.length - 1; i += 2) {
        swap(arr, i, i + 1);
    }

    for (int i = 0; i < arr.length; i++) {
        System.out.print(arr[i] + " ");
    }
}

public static void swap(int[] arr, int x, int y) {
    int temp = arr[x];
    arr[x] = arr[y];
    arr[y] = temp;
}
```

i = 6