⇒ Sorting

↳ Arrays.sort(arr_name);

```
public static void main(String[] args) {
    int[] arr = {4, 7, 1, -1, 0, 9 , 6};
    Arrays.sort(arr);

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

↳ primitive data type

↳ Arrays.sort(arr_name, Collections.reverseOrder());

```
public static void main(String[] args) {
    Integer[] arr = {4, 7, 1, -1, 0, 9 , 6};

    Arrays.sort(arr, Collections.reverseOrder());

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

→ Custom Sort

↳ concept which is used to modify a given function's logic accordingly.

↳ Comparator

↳ Comparable

```java
public static void main(String[] args) {
    Integer[] arr = {4, 7, 1, -1, 0, 9 , 6};

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

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

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

Implimentation

myComparator

Java comparator

class

fn

logic

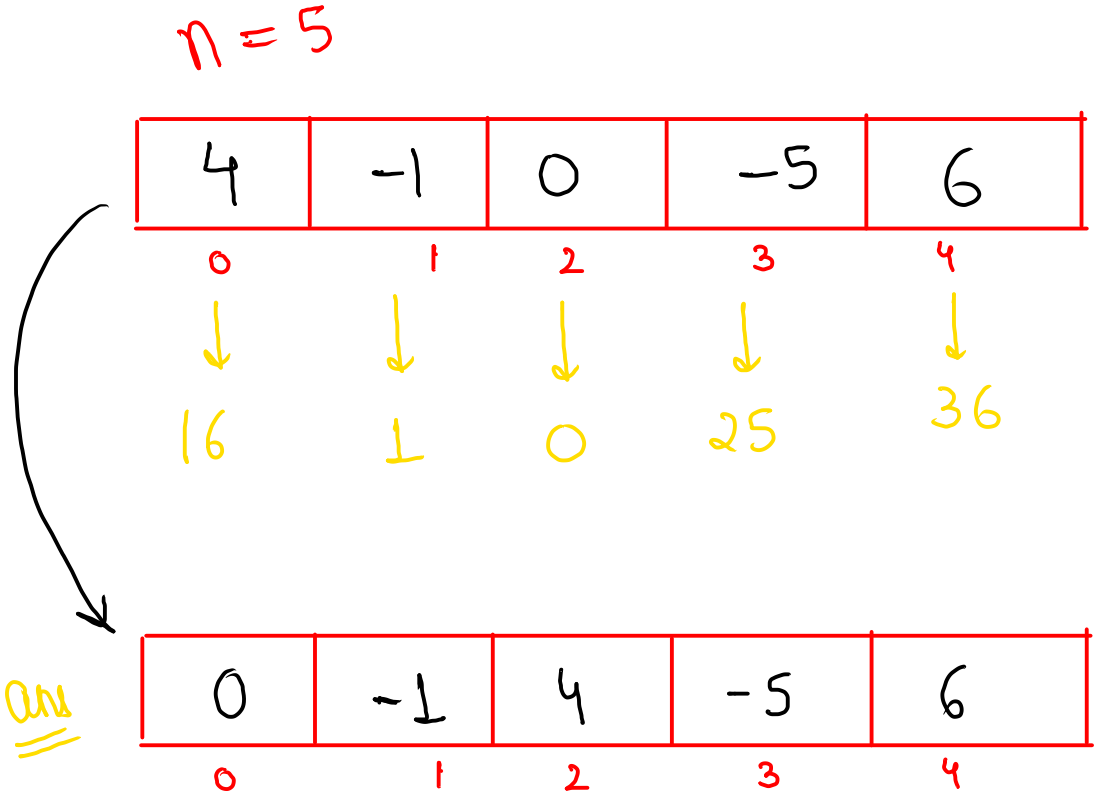
↳ lambda function

```java
public static void main(String[] args) {
    Integer[] arr = {4, 7, 1, -1, 0, 9 , 6};

    Arrays.sort(arr, (a, b) -> {          // lambda function
        // return a - b;    // increasing order
        return b - a;    // decrasing order
    });

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

Sort the array according to their Square of each element



Time :- Arrays.sort(arr) → O(N log N)

N → size of array

↳ using lambda function

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

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

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

↳ using comparator & compaiable

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

    //main logic
    Arrays.sort( arr, new myComparator() );

    // print
    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;
    }
}
```

Sort Array By Parity

a = even, b = even

a = odd, b = odd

a = odd, b = even

a = even, b = odd


```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 ) {   // both are even
            return a - b;   // increasing order
        } else if ( a % 2 != 0 && b % 2 != 0 ) {    // both are odd
            return a - b;    // increasing order
        } else if ( a % 2 == 0 && b % 2 != 0 ) {    // a == even, b == odd
            return -1;  // bring A first
        } else {        // a == odd, b == even
            return 1;       // bring B first
        }

    });

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